PCT/JP2004/019181

日　本　国　特　許　庁
JAPAN PATENT OFFICE

22.12.2004

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

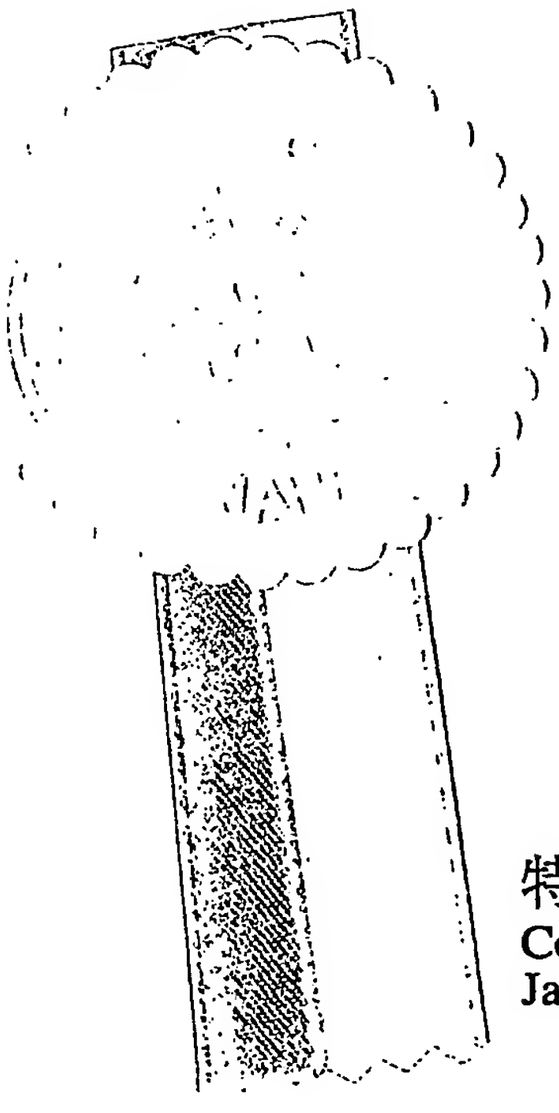
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出　願　年　月　日
Date of Application:　２００３年１２月２５日

出　願　番　号
Application Number:　特願２００３－４３１２４８

[ST. 10/C]:　[Ｊ Ｐ ２００３－４３１２４８]

出　願　人
Applicant(s):　古庄　晋二



２００５年　２月１０日

特許庁長官
Commissioner,
Japan Patent Office　小　川　洋

出証番号　出証特２００５－３００９１２５

【書類名】　特許願
【整理番号】　PK030079
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/30
　　G06F 15/00
【発明者】
　【住所又は居所】　神奈川県横浜市神奈川区松見町４丁目１１０１番地７　コートハウス菊名８０４号
　【氏名】　古庄　晋二
【特許出願人】
　【識別番号】　598108515
　【氏名又は名称】　古庄　晋二
【代理人】
　【識別番号】　100099715
　【弁理士】
　【氏名又は名称】　吉田　聡
【手数料の表示】
　【予納台帳番号】　231855
　【納付金額】　21,000円
【その他】　国等の委託研究の成果に係る特許出願（平成１４年度　情報処理振興事業協会　平成１４年度未踏ソフトウェア創造事業「超高速ＤＢアルゴリズムの超並列環境下のシミュレータ開発」に関する委託契約）
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1

【書類名】特許請求の範囲
【請求項１】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法であって、
各処理モジュールが、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、
各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信するステップと、
各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを１ずつ増やすステップと、
を有する情報処理方法。
【請求項２】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法であって、
各処理モジュールが、自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを前記情報処理システム内の他の処理モジュールへ送信するステップと、
各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値と該値の個数の対のリストである少なくとも一つの第２のリストを受信するステップと、
各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを、前記第２のリスト中の前記一致した値に対応した該値の個数分ずつ増やすステップと、
を有する情報処理方法。
【請求項３】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法であって、
各処理モジュールが、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、
各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信するステップと、
各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、各処理モジュールが、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やすステップと、
を有する情報処理方法。
【請求項４】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法であって、
各処理モジュールが、自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステ

ップと、
　各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値と該値の対のリストである少なくとも一つの第２のリストを受信するステップと、
　各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
　前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、各処理モジュールが、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを、前記第２のリスト中の前記値に対応した前記値の個数分ずつ増やすステップと、
を有する情報処理方法。
【請求項５】
　値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法であって、
　各処理モジュールが、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、
　各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信するステップと、
　各処理モジュールが、前記第２のリスト中の値が前記第１のリスト中に存在する場合に、前記第２のリスト中の前記値を消去し、二つ以上の前記第２のリスト中に同じ値が重複して出現する場合に、後から出現した方の第２のリスト中の該値を消去するステップと、
　各処理モジュールが、前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合に、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やすステップと、
を有する情報処理方法。
【請求項６】
　前記各処理モジュールは、情報の項目に対応した項目値を含むレコードの配列として表される表形式データを、項目値に対応した項目値番号の順序に該項目値が格納されている値リスト、及び、レコードの順番に項目値番号を指定する情報が格納されたポインタ配列の形式でメモリに記憶し、
　前記値のリストは前記表形式データを構成する前記値リストである、
請求項１乃至５のうちいずれか１項に記載の情報処理方法。
【請求項７】
　値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムであって、
　各処理モジュールは、
　自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
　他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する手段と、
　前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
　前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを１ずつ増やす手段と、
を有する、
情報処理システム。
【請求項８】
　値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムであって、

各処理モジュールは、
自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを前記情報処理システム内の他の処理モジュールへ送信する手段と、
他の処理モジュールから自処理モジュールへ送信された値と該値の個数の対のリストである少なくとも一つの第２のリストを受信する手段と、
前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、前記第１のリスト中の前記一致した値に対応したカウンタを、前記第２のリスト中の前記一致した値に対応した該値の個数分ずつ増やす手段と、
を有する、
情報処理システム。
【請求項９】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムであって、
各処理モジュールは、
自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する手段と、
前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やす手段と、
を有する、
情報処理システム。
【請求項１０】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムであって、
各処理モジュールは、
自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
他の処理モジュールから自処理モジュールへ送信された値と該値の対のリストである少なくとも一つの第２のリストを受信する手段と、
前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを、前記第２のリスト中の前記値に対応した前記値の個数分ずつ増やす手段と、
を有する、
情報処理システム。
【請求項１１】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムであって、
各処理モジュールは、
自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一

つの第２のリストを受信する手段と、
　前記第２のリスト中の値が前記第１のリスト中に存在する場合に、前記第２のリスト中の前記値を消去し、二つ以上の前記第２のリスト中に同じ値が重複して出現する場合に、後から出現した方の第２のリスト中の該値を消去する手段と、
　前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合に、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やす手段と、
を有する、
情報処理システム。
【請求項１２】
　前記各処理モジュールは、情報の項目に対応した項目値を含むレコードの配列として表される表形式データを、項目値に対応した項目値番号の順序に該項目値が格納されている値リスト、及び、レコードの順番に項目値番号を指定する情報が格納されたポインタ配列の形式で記憶するメモリを具備し、
　前記値のリストは前記表形式データを構成する前記値リストである、
請求項７乃至１１のうちいずれか１項に記載の情報処理システム。
【請求項１３】
　値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムにおいて、
　各処理モジュールのコンピュータに、
　自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する機能と、
　他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する機能と、
　前記第２のリスト中の値と、前記第１のリスト中の値を比較する機能と、
　前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを１ずつ増やす機能と、
を実現させるためのプログラム。
【請求項１４】
　値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムにおいて、
　各処理モジュールのコンピュータに、
　自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを前記情報処理システム内の他の処理モジュールへ送信する機能と、
　他の処理モジュールから自処理モジュールへ送信された値と該値の個数の対のリストである少なくとも一つの第２のリストを受信する機能と、
　前記第２のリスト中の値と、前記第１のリスト中の値を比較する機能と、
　前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、前記第１のリスト中の前記一致した値に対応したカウンタを、前記第２のリスト中の前記一致した値に対応した該値の個数分ずつ増やす機能と、
を実現させるためのプログラム。
【請求項１５】
　値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムにおいて、
　各処理モジュールのコンピュータに、
　自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する機能と、

他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する機能と、
前記第２のリスト中の値と、前記第１のリスト中の値を比較する機能と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やす機能と、
を実現させるためのプログラム。
【請求項１６】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムにおいて、
各処理モジュールのコンピュータに、
自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する機能と、
他の処理モジュールから自処理モジュールへ送信された値と該値の対のリストである少なくとも一つの第２のリストを受信する機能と、
前記第２のリスト中の値と、前記第１のリスト中の値を比較する機能と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを、前記第２のリスト中の前記値に対応した前記値の個数分ずつ増やす機能と、
を実現させるためのプログラム。
【請求項１７】
値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムにおいて、
各処理モジュールのコンピュータに、
自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する機能と、
他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する機能と、
前記第２のリスト中の値が前記第１のリスト中に存在する場合に、前記第２のリスト中の前記値を消去し、二つ以上の前記第２のリスト中に同じ値が重複して出現する場合に、後から出現した方の第２のリスト中の該値を消去する機能と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合に、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やす機能と、
を実現させるためのプログラム。
【請求項１８】
前記各処理モジュールは、情報の項目に対応した項目値を含むレコードの配列として表される表形式データを、項目値に対応した項目値番号の順序に該項目値が格納されている値リスト、及び、レコードの順番に項目値番号を指定する情報が格納されたポインタ配列の形式で記憶するメモリを具備し、
前記値のリストは前記表形式データを構成する前記値リストである、
請求項１３乃至１７のうちいずれか１項に記載のプログラム。
【請求項１９】
請求項１３乃至１８のうちいずれか１項記載のプログラムを記録したコンピュータ読み取り可能な記録媒体。

【書類名】明細書
【発明の名称】情報処理システム及び情報処理プログラム
【技術分野】
【０００１】
本発明は、大量のデータを処理する情報処理方法及び情報処理装置に係わり、特に、並列コンピュータのアーキテクチャを採用した情報処理方法及び情報処理システムに関する。
【背景技術】
【０００２】
従来、大量の情報を蓄積し、蓄積された情報を検索、集計するデータ処理が行われている。これらのデータ処理は、例えば、ＣＰＵと、メモリと、周辺機器インタフェースと、ハードディスクのような補助記憶装置と、ディスプレイ及びプリンタのような表示装置と、キーボード及びマウスのような入力装置と、電源ユニットとがバスを介して接続された周知のコンピュータシステムで使用され、特に、市場で容易に入手可能なコンピュータシステムで動作可能なソフトウェアとして提供される。上記の検索・集計等のデータ処理を行うため、特に、大量のデータを蓄積する各種データベースが知られている。大量のデータの中でも、特に、表形式で表現され得るデータを処理したいという要求は強い。
【０００３】
大量のデータを効率よく検索、集計できるか否かは、大量のデータを格納する形式に依存する。従来、一般的な格納技術として、所謂「行単位」格納技術及び「項目単位」格納技術が知られている。行単位格納技術の場合、レコード番号毎に構成された性別、年齢及び職業の項目値の組がレコード番号順に、論理的アドレスが増加する順番でディスク上に格納されている。一方、項目単位格納技術の場合、項目毎にレコード番号順に、項目値が、論理的アドレスが増加する方向にディスク上に格納されている。
【０００４】
上記の従来技術の場合、全てのレコード番号の全ての項目に対する項目値が、そのまま、（レコード番号の１次元と、それ以外の項目値の１次元とからなる）２次元のデータ構造に格納されている。以下、このようなデータ構造を特に「データ表」と称する。従来技術の場合、蓄積されたデータを検索、集計する際には、このデータ表をアクセスすることにより行われている。
【０００５】
また、項目の値をそのまま項目値として格納する方法の他に、値をコード変換して、項目値としてコードを格納する方法も知られている。この場合でも、コード変換されたコードが項目値としてデータ表に格納されている点に変わりはない。
【０００６】
上記従来技術のデータ表タイプのデータ構造を用いて格納された大量のデータを検索、集計する場合、このようなデータ表をアクセスするためのアクセス時間に起因して検索、集計の処理時間が長くなるという問題点がある。
【０００７】
また、データ表には少なくとも以下に述べる本質的な欠点がある。
（１）データ表はサイズが巨大化しやすく、しかも例えば項目毎等に（物理的に）分割することが困難である。実際上、集計・検索のためにデータ表をメモリなどの高速な記憶装置上に展開することは困難である。
（２）データ表は、各項目値を同時にソートした形で保持できない。
（３）データ表には、同一値が何度も出現する。
【０００８】
これに対して、本願発明者は、大量のデータの検索・集計のスピードを大幅に向上させるため、従来のデータ表の機能を有すると共に、データ表に基づくデータ構造の上記問題点が解決されたデータ管理機構を設けることにより、表形式データを検索・集計・ソートする方法及びその方法を実施する装置を提案している（例えば、特許文献１を参照。）。

【０００９】
提案された表形式データを検索・集計する方法及び装置は、通常のコンピュータシステムで使用可能な新規のデータ管理機構を導入する。このデータ管理機構は原則として値管理テーブルと、値管理テーブルへのポインタ配列とを有する。

【００１０】
図１は、従来のデータ管理機構の説明図である。同図には、値管理テーブル１１０と値管理テーブルへのポインタ配列１２０とが示されている。値管理テーブル１１０とは、表形式データの各項目に対し、その項目に属する項目値が順序付け（整数化）された項目値番号の順番に、上記項目値番号に対応した項目値（符号１１１参照）と、上記項目値に関連した分類番号（符号１１２参照）とが格納されたテーブルである。値管理テーブルへのポインタ配列１２０とは、表形式データのある列（即ち、項目）の項目値番号、即ち値管理テーブル１１０へのポインタが表形式データのレコード番号順に格納された配列である。

【００１１】
値管理テーブルへのポインタ配列１２０と値管理テーブル１１０とを組み合わせることにより、あるレコード番号が与えられたとき、所定の項目に関する値管理テーブルへのポインタ配列１２０からそのレコード番号に対応して格納された項目値番号を取り出し、次に、値管理テーブル１１０内でその項目値番号に対応して格納された項目値を取り出すことにより、レコード番号から項目値を得ることができる。したがって、従来のデータ表と同様に、レコード番号（行）と項目（列）という座標を用いて全てのデータ（項目値）を参照することができる。

【００１２】
このように表形式データの項目中のある項目に対して生成された値管理テーブルと、値管理テーブルへのポインタ配列とを含むデータ管理機構は、以下の説明では、特に情報ブロックと呼ぶ場合がある。

【００１３】
従来のデータ表は、レコードに対応した行と、項目に対応した列とからなる座標を用いて全てのデータを一体的に管理しているのに対して、この情報ブロックは、表形式の列、即ち、項目毎にデータを完全に分離している点に特徴がある。このデータ管理機構によれば、大量のデータが項目毎に分離されているので、検索・集計に必要な項目に関するデータのみをメモリ等の高速な記憶装置上に取り込むことが可能であり、その結果としてデータへのアクセス時間が短縮されるので、検索・集計の処理が高速化され、項目数が非常に多いデータの場合でも、パフォーマンスを低下させることなく取り扱えるようになる。

【００１４】
また、この情報ブロックの場合、項目値は値管理テーブルに格納され、値が存在する位置を示すレコード番号は値管理テーブルへのポインタ配列に関連付けられているので、項目値がレコード番号順に並べられている必要がない。したがって、検索・集計に適するようにデータを項目値に関してソートすることができるようになる。これにより、目的の値と一致する項目値がデータ中に存在するか否かの判定が高速に行えるようになる。その上、項目値は項目値番号と対応しているので、項目値が長いデータや文字列等であっても整数として取り扱うことができる。

【００１５】
さらに、このデータ管理機構によれば、値管理テーブル１１０の全ての項目値番号は異なる項目値と対応しているので、特定の値を有する項目値を含むレコードを抽出する場合に必要とされる特定の値と項目値との比較の回数は、最大で項目値の種類の数、即ち、項目値番号の個数であり、比較演算の回数が著しく低減され、検索・集計の高速化が図られる。その際には、ある項目値が該当するかどうかを調べた結果を格納する場所が必要であるが、例えば分類番号１１２をその格納場所として使用することができる。

【００１６】
図２には、項目値を格納した項目値配列２１１と、分類番号を格納した分類番号配列２

１２と、存在数を格納した存在数配列２１４とからなる値管理テーブル２１０を含む情報ブロックが示されている。存在数配列２１４には、ある項目に関する各項目値が全データ中に何個ずつ存在するかを示す数、換言すれば、所定の項目値を有するレコードの個数が格納される。このような存在数配列２１４を値管理テーブル２１０に準備しておくことにより、検索・ソート・集計の際に必要とされる「どのようなデータが（いくつ）存在するか？」、「このデータは上から何番目のデータであるか？」、或いは、「上から○○番目のデータは何か？」というような情報を直ちに得ることができるようになり、検索・ソート・集計の高速化が図れる。

【００１７】

しかし、このようなデータ管理機構においても、レコード数が増大するのにしたがって、上記値リストやポインタ配列、特に、ポインタ配列は非常に大きくなるが、処理可能なデータ量は、利用されるハードウェア資源によって制限される。

【００１８】

大規模データの処理は、上記のような表形式データの情報処理以外の分野でも要求されている。社会全体のさまざまな場所にコンピュータが導入され、インターネットをはじめとするネットワークが浸透した今日では、そこここで、大規模なデータが蓄積されるようになった。このような大規模データを処理するには、膨大な計算が必要で、そのために並列処理を導入しようと試みるのは自然である。

【００１９】

並列処理アーキテクチャは「共有メモリ型」と「分散メモリ型」に大別される。前者（「共有メモリ型」）は、複数のプロセッサが１つの巨大なメモリ空間を共有する方式である。この方式では、プロセッサ群と共有メモリ間のトラフィックがボトルネックとなるので、百を越えるプロセッサを用いて現実的なシステムを構築することは容易ではない。したがって、例えば１０億個の浮動小数点変数の平方根を計算する際、単一ＣＰＵに対する加速比は、せいぜい１００倍ということになる。経験的には、３０倍程度が上限である。

【００２０】

後者（「分散メモリ型」）は、各プロセッサがそれぞれローカルなメモリを持ち、これらを結合してシステムを構築する。この方式では、数百～数万ものプロセッサを組み込んだハードウェアシステムの設計が可能である。したがって、上記１０億個の浮動小数点変数の平方根を計算する際の単一ＣＰＵに対する加速比を、数百～数万倍とすることが可能である。

【特許文献１】国際公開第ＷＯ００／１０１０３号パンフレット

【発明の開示】

【発明が解決しようとする課題】

【００２１】

しかしながら、「分散メモリ型」の並列処理アーキテクチャにもいくつかの課題が存在する。

【００２２】

[第１の課題：巨大配列の分掌管理]

「分散メモリ型」の第１の課題は、データの分掌管理の問題である。

【００２３】

巨大なデータ（一般的には配列なので、以降、配列で説明する）は、１つのプロセッサの所有するローカルメモリに収容できるものではなく、必然的に複数のローカルメモリに分掌管理される。効率的かつ柔軟な分掌管理メカニズムを導入しないと、プログラムの開発および実行に際してさまざまな障害を抱え込むことになることは明らかである。

【００２４】

[第２の課題：プロセッサ間通信の効率の低さ]

分散メモリ型システムの各プロセッサが、巨大配列にアクセスしようとすると、自己の所有するローカルメモリ上の配列要素に対しては速やかにアクセスできるものの、他のプロセッサが所有する配列要素へのアクセスはプロセッサ間通信を必須とする。このプロセ

ッサ間通信はローカルメモリとの通信に比べ、極端にパフォーマンスが低く、最低でも１００クロックかかると言われている。このため、ソート実施時には、巨大配列全域にわたる参照が実施され、プロセッサ間通信が多発するため、パフォーマンスが極端に低下する。

【００２５】

この問題点につき、より具体的に説明を加える。１９９９年現在、パソコンは、１～数個のＣＰＵを用いて、「共有メモリ型」として構成されている。このパソコンに使用される標準的なＣＰＵは、メモリバスの５～６倍程度の内部クロックで動作し、その内部に自動的な並列実行機能やパイプライン処理機能が装備されており、およそ1データを１クロック（メモリバス）で処理できる。

このため、「分散メモリ型」のマルチプロセッサシステムでは、プロセッサ数は多いのに、シングルプロセッサ（共有メモリ型）よりも１００倍遅くなることになりかねない。

【００２６】

[第３の課題：プログラムの供給]

「分散メモリ型」の第３の課題は、多数のプロセッサにどうやってプログラムを供給するか、という問題である。

非常に多数のプロセッサに、別々のプログラムをロードし、全体を協調動作させる方式（ＭＩＭＤ：Multiple Instruction Stream, Multiple Data Stream）では、プログラムの作成、コンパイル、配信のために多大な負荷を要する。

その一方、多数のプロセッサを同一のプログラムで動作させる方式（ＳＩＭＤ：Single Instruction Stream, Multiple Data Stream）では、プログラムの自由度が減少し、所望の結果をもたらすプログラムが開発できない事態も想定される。

【００２７】

したがって、上記の従来の分散メモリ型の並列アーキテクチャに基づく情報処理技術では、プロセッサ間通信ができるだけ少なくなるように、大規模データをプロセッサ間で共有することなく、大規模データを個々のプロセッサ内に保持したまま、大規模データの処理を実現することが求められている。

【００２８】

そこで、本発明は、並列コンピュータのアーキテクチャを採用して大量のデータを情報処理する際に、複数のプロセッサ間でのデータ処理を少ない通信量で高速に実現するための情報処理方法の提供を目的とする。

更に、本発明は、上記の情報処理方法を実現する情報処理システムの提供を目的とする。

また、本発明は、上記の情報処理方法を実現するためコンピュータによって実行されるプログラムの提供を目的とする。

【課題を解決するための手段】

【００２９】

本発明は、表形式データの実体的要素である値リスト及びポインタ配列を個々の処理モジュールにローカルに保存し、複数の処理モジュール間では、データ自体ではなく、データの順序番号（又は、順位）という指標がグローバルに保持されるという分散メモリ型の並列処理アーキテクチャを採用している。また、本発明は、単一命令により種々のメモリに記憶されたデータを入出力し処理するように、処理と通信が統合されたアルゴリズムを採用している。

【００３０】

上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法は、請求項１に記載されるように、

各処理モジュールが、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、

各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信するステップと、
各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを１ずつ増やすステップと、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値を処理モジュール間で相互に交換することによって、マッチングした値の出現数を計数することが可能になる。
【００３１】
また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法は、請求項２に記載されるように、
各処理モジュールが、自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを前記情報処理システム内の他の処理モジュールへ送信するステップと、
各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値と該値の個数の対のリストである少なくとも一つの第２のリストを受信するステップと、
各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを、前記第２のリスト中の前記一致した値に対応した該値の個数分ずつ増やすステップと、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値、及び、各処理モジュール内に存在するそれらの値の個数を処理モジュール間で相互に交換することによって、マッチングした値の出現数を計数する際に、データ交換に要する通信量を削減することが可能になる。
【００３２】
また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法は、請求項３に記載されるように、
各処理モジュールが、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、
各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信するステップと、
各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、各処理モジュールが、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やすステップと、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値を処理モジュール間で相互に交換することによって、累計数を算出し、値を順序付けることが可能になる。
【００３３】
また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法は、

請求項４に記載されるように、
　各処理モジュールが、自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、
　各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値と該値の対のリストである少なくとも一つの第２のリストを受信するステップと、
　各処理モジュールが、前記第２のリスト中の値と、前記第１のリスト中の値を比較するステップと、
　前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、各処理モジュールが、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを、前記第２のリスト中の前記値に対応した前記値の個数分ずつ増やすステップと、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値、及び、各処理モジュール内に存在するそれらの値の個数を処理モジュール間で相互に交換することによって、累計数を算出し、値を順序付ける際に、データ交換に要するデータ量を削減することが可能になる。
【００３４】
　また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムにおいて、複数の処理モジュール間で相互にデータを通信し処理する情報処理方法は、請求項５に記載されるように、
　各処理モジュールが、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信するステップと、
　各処理モジュールが、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信するステップと、
　各処理モジュールが、前記第２のリスト中の値が前記第１のリスト中に存在する場合に、前記第２のリスト中の前記値を消去し、二つ以上の前記第２のリスト中に同じ値が重複して出現する場合に、後から出現した方の第２のリスト中の該値を消去するステップと、
　各処理モジュールが、前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合に、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やすステップと、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値を処理モジュール間で相互に交換することによって、それらの値に複数の処理モジュール間で共通の順序番号を付与することが可能になる。
【００３５】
　請求項１乃至５に記載された情報処理方法において、請求項６に記載されるように、
　前記各処理モジュールは、情報の項目に対応した項目値を含むレコードの配列として表される表形式データを、項目値に対応した項目値番号の順序に該項目値が格納されている値リスト、及び、レコードの順番に項目値番号を指定する情報が格納されたポインタ配列の形式でメモリに記憶し、前記値のリストは前記表形式データを構成する前記値リストである。これにより、リスト内のデータが予め昇順若しくは降順に並べられるので、比較のための演算を高速に行うことが可能になる。
【００３６】
　また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムは、請求項７に記載されるように、
　各処理モジュールが、
　自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理

システム内の他の処理モジュールへ送信する手段と、
　他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する手段と、
　前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
　前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、各処理モジュールが、前記第１のリスト中の前記一致した値に対応したカウンタを１ずつ増やす手段と、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値を処理モジュール間で相互に交換することによって、マッチングした値の出現数を計数することが可能になる。
【００３７】
　また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムは、請求項８に記載されるように、
　各処理モジュールが、
　自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを前記情報処理システム内の他の処理モジュールへ送信する手段と、
　他の処理モジュールから自処理モジュールへ送信された値と該値の個数の対のリストである少なくとも一つの第２のリストを受信する手段と、
　前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
　前記第２のリスト中の値が前記第１のリスト中の値と一致した場合に、前記第１のリスト中の前記一致した値に対応したカウンタを、前記第２のリスト中の前記一致した値に対応した該値の個数分ずつ増やす手段と、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値、及び、各処理モジュール内に存在するそれらの値の個数を処理モジュール間で相互に交換することによって、マッチングした値の出現数を計数する際に、データ交換に要する通信量を削減することが可能になる。
【００３８】
　また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムは、請求項９に記載されるように、
　各処理モジュールが、
　自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
　他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する手段と、
　前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
　前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やす手段と、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値を処理モジュール間で相互に交換することによって、累計数を算出し、値を順序付けることが可能になる。
【００３９】
　また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムは、請求項１０に記載されるように、
　各処理モジュールが、

自処理モジュールのメモリに格納されている値と該値の個数の対のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
他の処理モジュールから自処理モジュールへ送信された値と該値の対のリストである少なくとも一つの第２のリストを受信する手段と、
前記第２のリスト中の値と、前記第１のリスト中の値を比較する手段と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを、前記第２のリスト中の前記値に対応した前記値の個数分ずつ増やす手段と、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値、及び、各処理モジュール内に存在するそれらの値の個数を処理モジュール間で相互に交換することによって、累計数を算出し、値を順序付ける際に、データ交換に要するデータ量を削減することが可能になる。

【００４０】

また、上記目的を達成するため、本発明によれば、値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールと、複数の処理モジュールを論理的に環状に接続する接続路と、を含み、複数の処理モジュール間で相互にデータを通信し処理する情報処理システムは、請求項１１に記載されるように、
各処理モジュールが、
自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信する手段と、
他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信する手段と、
前記第２のリスト中の値が前記第１のリスト中に存在する場合に、前記第２のリスト中の前記値を消去し、二つ以上の前記第２のリスト中に同じ値が重複して出現する場合に、後から出現した方の第２のリスト中の該値を消去する手段と、
前記第２のリスト中の値よりも後に順位付けされる値が前記第１のリスト中に存在する場合に、前記第２のリスト中の値の直後に順位付けされる前記第１のリスト中の値に対応したカウントを１ずつ増やす手段と、
を有する。これにより、複数の処理モジュール間に重複のある整数、文字列、浮動小数点などの値が分散して存在している場合に、それらの値を処理モジュール間で相互に交換することによって、それらの値に複数の処理モジュール間で共通の順序番号を付与することが可能になる。

【００４１】

請求項７乃至１１に記載された情報処理方法において、請求項１２に記載されるように、
前記各処理モジュールは、情報の項目に対応した項目値を含むレコードの配列として表される表形式データを、項目値に対応した項目値番号の順序に該項目値が格納されている値リスト、及び、レコードの順番に項目値番号を指定する情報が格納されたポインタ配列の形式で記憶するメモリを具備し、前記値のリストは前記表形式データを構成する前記値リストである。これにより、リスト内のデータが予め昇順若しくは降順に並べられるので、比較のための演算を高速に行うことが可能になる。

【００４２】

更に、上記目的を達成するため、本発明のプログラムは、請求項１３乃至１８に記載されているように、上記本発明の情報処理方法の各ステップをコンピュータに実行させ、或いは、上記本発明の情報処理システムの各機能をコンピュータに実現させる。これにより、本発明の様々な機能をコンピュータに実現させるためのプログラムを提供することができる。このプログラムは、通信回線や記録媒体を用いてコンピュータに提供することが可能である。

【００４３】

更に、請求項１８に係る発明は、請求項１３乃至１７のうちいずれか１項記載のプログラムを記録したコンピュータ読み取り可能な記録媒体を提供する。

【発明の効果】

【００４４】

本発明によれば、分散メモリ型の並列処理アーキテクチャに基づいて、新規のデータ構造及び並列処理アルゴリズムを採用することにより、著しく高速な並列処理を実現可能な情報処理方法及び情報処理システムを提供することが可能となる。

【発明を実施するための最良の形態】

【００４５】

[ハードウェア構成]

以下、添付図面を参照して、本発明の実施の形態につき説明を加える。図３は、本発明の実施の形態にかかる情報処理システムの概略を示すブロックダイヤグラムである。この実施形態では、処理モジュールは、プロセッサ付きメモリモジュール（以下、「ＰＭＭ」と称する）により構成される。図３に示すように、この実施の形態においては、複数の処理モジュールを論理的に環状に接続するため、複数のプロセッサ付きメモリモジュールＰＭＭ３２－０、ＰＭＭ３２－１、ＰＭＭ３２－２、・・・がリング状に配置され、隣接するメモリモジュール間を、時計回りにデータを伝達する第１のバス（たとえば、符号３４－０、３４－１参照）、および、反時計回りにデータを伝達する第２のバス（たとえば、符号３６－０、３６－１参照）が接続している。第１のバスおよび第２のバスでは、ＰＭＭ間のパケット通信が実行される。本実施の形態において、このパケット通信が実行される伝送路（パケット伝送路）を、第１のバスおよび第２のバスと称する。

【００４６】

本実施の形態では、ＰＭＭを、一方が時計回りにパケットを伝送する第１のバス（第１の伝送路）、他方が反時計回りにパケットを伝送する第２のバス（第２の伝送路）にて、リング状に接続している。このような構成は、パケット伝送の遅延時間などを均一化することができるため有利である。

【００４７】

尚、処理モジュール間の物理的な接続形態は、本実施の形態に示されるような形態に限定されるものではなく、処理モジュールを論理的に環状に接続できる形態であればどのような形態でもよく、例えば、バス型、スター型などの種々の接続形態を採用することができる。

【００４８】

図４は、ＰＭＭ３２の構造の一例を示す図である。図４に示すように、各ＰＭＭ３２－ｉは、ＰＭＭ間で共通の命令にしたがって、メモリのアクセス、演算の実行などを制御する制御回路４０と、バスインタフェース（Ｉ／Ｆ）４２と、メモリ４４とを備えている。

メモリ４４は、複数のバンクＢＡＮＫ０、１、・・・、ｎ（符号４６－０、・・・、ｎ）を有し、それぞれに、後述する所定の配列を記憶できるようになっている。また、制御回路４０は、外部の他のコンピュータ等とのデータ授受が可能である。また、他のコンピュータが、バスアービトレーションにより、メモリの所望のバンクにアクセスできるようにしても良い。

【００４９】

[処理の対象]

本実施の形態における情報処理の一例は集計処理である。集計とは、例えば、情報の項目に対応した項目値を含むレコードの配列として表される表形式データから、ある項目（次元）のある項目値（次元値）毎に、別の項目の項目値（メジャー）を集計することである。メジャーの集計とは、メジャーの個数をカウントしたり、メジャーの総和を算出したり、メジャーの平均値を算出したりすることである。また、次元数は、２次元以上でも構わない。例えば、図５は、ある保育園における園児の性別・年齢・身長の論理的な表形式データである。ここで、性別の人数を求める処理や、性別・年齢別に身長の合計値を求める処理は、本実施の形態における情報処理の一例としての集計処理である。

【００５０】
［従来のデータの記憶構造］
図５に示された表形式データは、上述の国際公開第ＷＯ００／１０１０３号に提案したデータ管理機構を用いることにより、単一コンピュータ内では図６に示されるようなデータ構造として記憶される。図５に示すように、表形式データの各レコードの並び順の番号と、内部データの並び順の番号を対応付ける配列６０１（以下、この配列を「OrdSet」のように略記する。）には、表形式の各レコード毎に内部データの並び順番号が値として配置される。この例では、すべての表形式データが内部データとして表されるため、表形式データのレコード番号と内部データの並び順番号とは一致する。
【００５１】
例えば、性別に関しては、表形式データのレコード０に対応する内部データの並び順番号は、配列OrdSet６０１から「０」であることがわかる。並び順番号が「０」であるレコードに関する実際の性別の値、即ち、「男」又は「女」は、実際の値が所定の順序に従ってソートされた値リスト６０３（以下、値リストを「VL」のように略記する。）へのポインタ配列６０２（以下、ポインタ配列を「VNo」のように略記する。）を参照することによって取得できる。ポインタ配列６０２は、配列OrdSet６０１に格納されている並び順番号の順に従って、実際の値リスト６０３中の要素を指し示すポインタを格納している。これにより、表形式データのレコード「０」に対応する性別の項目値は、（１）配列OrdSet６０１からレコード「０」に対応する並び順番号「０」を取り出し、（２）値リストへのポインタ配列６０２から並び順番号「０」に対応する要素「１」を取り出し、（３）値リスト６０３から、値リストへのポインタ配列６０２から取り出された要素「１」によって指し示される要素「女」を取り出すことにより取得できる。
他のレコードに対しても、また、年齢及び身長に関しても同様に項目値を取得することができる。
【００５２】
このように表形式データは、値リストVLと、値リストへのポインタ配列VNoの組合せにより表現され、この組合せを、特に、「情報ブロック」とも称する。図６には、性別、年齢及び身長に関する情報ブロックがそれぞれ情報ブロック６０８、６０９及び６１０として示されている。
【００５３】
単一のコンピュータが、単一のメモリ（物理的には複数であっても良いが、単一のアドレス空間に配置されアクセスされるという意味で単一のメモリ）であれば、当該メモリに、順序集合の配列OrdSet、各情報ブロックを構成する値リストVLおよびポインタ配列VNoとを記憶しておけばよい。しかしながら、大量のレコードを保持するためには、その大きさに伴ってメモリ容量も大きくなるため、これらを分散配置できるのが望ましい。また、処理の並列化の観点からも、分散配置された情報を分掌把握できるのが望ましい。そこで、本実施の形態においては、複数のＰＭＭが、重なることなくレコードのデータを分掌把握し、ＰＭＭ同士のパケット通信により、高速な集計を実現している。
【００５４】
［本実施の形態によるデータ記憶構造］
図７は、本実施の形態によるデータ記憶構造の説明図である。同図では、図５及び図６によって示された表形式のデータが、一例として、ＰＭＭ－０、ＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３の４個の処理モジュールに分散配置され、分掌把握されている。説明の便宜上、処理モジュールの個数は４個であるが、本発明は処理モジュールの個数によって限定されるものではない。
【００５５】
本実施の形態では、各ＰＭＭで分掌把握されているレコードを、ＰＭＭ－０からＰＭＭ－３までの４個のＰＭＭで掌握されているすべてのレコードの中で一意に順序付けることができるようにするため、各レコードにはグローバル・レコード番号が一意に割り当てられている。図７では、グローバル・レコード番号は「GOrd」として表されている。このグ

ローバル・レコード番号GOrdは、各ＰＭＭ内の配列OrdSetの各要素が、すべてのレコード中で何番目であるかを示している。ここで、配列OrdSetは、データ全体から各ＰＭＭ内部への順序保存写像となるように定められているので、GOrdは昇順にすることができる。また、各ＰＭＭ内において、GOrd配列（＝グローバル順序集合配列）のサイズはOrdSet配列（順序集合配列）のサイズと一致している。

【００５６】

更に、本実施の形態では、各ＰＭＭで分掌把握されている項目値、即ち、値リストVL中の各値が、すべてのＰＭＭで掌握されている項目値の中で何番目の位置にあるかを示すためのグローバル項目値番号が設けられる。図７では、このグローバル項目値番号は「GVNo」として示されている。値リストVLは値の順（例えば、昇順又は降順）に並べられているので、グローバル項目値番号GVNoも昇順（又は降順）に設定される。配列GVNoのサイズは、配列VLのサイズと一致している。各処理モジュールで個別に掌握されている項目値が全体の中で何番目であるかを識別することにより、各処理モジュールでの集計結果を全体として一つに統合することが可能になる。

【００５７】

尚、図７において、各ＰＭＭに割り当てられている値OFFSETは、当該ＰＭＭが分掌する先頭のレコードが、図６に示された一体的なレコードの中の何番目のレコードに対応しているかを示すためのオフセット値である。上述のように、各ＰＭＭの配列OrdSetは、データ全体から各ＰＭＭ内部への順序保存写像となるように定められているので、このオフセット値OFFSETと当該ＰＭＭにおける配列OrdSetの要素の値を合計した値は、グローバル・レコード番号GOrdと一致する。好ましくは、このオフセット値が各ＰＭＭに通知され、各ＰＭＭはこのオフセット値OFFSETに基づいてグローバル・レコード番号を決定することができる。

【００５８】

各ＰＭＭのグローバル・レコード番号GOrd及びグローバル項目値番号GVNoは、予め各ＰＭＭの外部で計算して各ＰＭＭに設定することができるが、後述のコンパイル処理によって各ＰＭＭ自体が設定することも可能である。

【００５９】

［グローバル集合配列Gordとグローバル項目値番号配列GNoについて］

次に、本実施の形態にて導入した配列GOrdおよび配列GVNoの意義について説明する。グローバル順序集合配列GOrdは、各ＰＭＭが掌握するローカルな表形式データを集合させたグローバルな表形式データ中、各ＰＭＭの掌握する表形式データの各レコードの位置（順位）を示している。即ち、本実施の形態では、グローバル順序集合配列GOrd及び順序集合配列OrdSetにより、レコードの位置情報を、グローバルな成分とローカルな成分とに分離し、これにより、グローバルな表形式データを扱うことが可能となるとともに、各ＰＭＭが単独で処理を実行することも可能となる。

【００６０】

以下の実施例の説明では、ＰＭＭが各項目の情報ブロックを保持するように構成されているが、ＰＭＭが表形式データをそのまま保持するような場合でも、上記GOrdは同様に機能する。

【００６１】

例えば、以下の実施例においてコンパイルが終了した状態で、グローバル順序集合配列GOrdの値の順序で、各項目の項目値を取り出していくことにより、表形式データ全体のビューを作成することができる。

【実施例１】

【００６２】

［集計処理］

次に、本実施の形態による集計処理を説明する。実施例１による集計アルゴリズムは、すべての処理モジュールで同じ処理を実行できるように構成されている。また、この集計アルゴリズムは、単一の集計処理命令を複数の処理モジュールへ与えることにより、複数

の処理モジュールが並列に動作して集計処理を実行できるように構成されている。すべての処理モジュールは、同じ動作を実行するので、一つのプログラムを作成するだけで、並列処理を実現できる。

【００６３】

実施例１の集計アルゴリズムは、集計のための次元値にすべての処理モジュール間で共通のグローバル次元値番号を付与し、各処理モジュール内で次元値番号毎にメジャーを集計し、最後に、グローバルに、即ち、すべての処理モジュール間で共通にメジャーを集計する。このため、実施例１の集計アルゴリズムによれば、値リスト、及び、値リストへのポインタ配列は、各処理モジュールでローカルに保持される。また、この集計アルゴリズムによれば、値リスト及びポインタ配列は、複数の処理モジュール間で共通に保持されるのではなく、次元値の順番という基準が複数の処理モジュール間でグローバルに保持される。その結果として、複数の処理モジュールが集計に必要なデータを取得するために相互にメモリへアクセスすることが回避され、次元値の順番を決定するために必要なデータだけが処理モジュール間で通信されるので、通信量が削減され処理の高速化が図られる。

【００６４】

図８は、実施例１による集計処理のフローチャートである。図８に示されるように、最初に、各処理モジュールに分掌管理された表形式データを準備する（ステップ８０１）。より具体的には、各処理モジュールは、自処理モジュール内のレコードに対して複数の処理モジュール間で一意に割り当てられたグローバル・レコード番号と、自処理モジュール内の項目値に対して複数の処理モジュール間で順序付けられたグローバル項目値番号とをメモリに記憶する。

【００６５】

次に、各処理モジュールは、自処理モジュール内で、少なくとも１次元以上の指定された項目のグローバル項目値番号の組の番号順にレコードをソートする（ステップ８０２）。

【００６６】

更に、各処理モジュールは、レコードに対応したグローバル項目値番号の組を、ソートされたレコードの順番に次元値番号を付与してメモリに格納する（ステップ８０３）。

【００６７】

次に、各処理モジュールは、他の処理モジュールからグローバル項目値番号の組を相互に取得し、自処理モジュール内のグローバル項目値番号の組よりも前に順序付けられる組の個数をカウントし、自処理モジュール内のグローバル項目値番号の組の次元値番号をカウントされた個数分だけ引き上げることにより、グローバル項目値番号の組に対して、複数の処理モジュール間で共通のグローバル次元値番号を付ける（ステップ８０４）。

【００６８】

続いて、各処理モジュールは、グローバル項目値番号の組毎に、所定の情報の項目の項目値を所定の規則に従って集計することによりローカル集計値を算出する（ステップ８０５）。最後に、各処理モジュールは、他の処理モジュールからグローバル項目値番号の組毎のローカル集計値を取得し、取得された集計値をグローバル項目値番号の組毎に集計することにより集計値を算出する（ステップ８０６）。

【００６９】

また、各処理モジュールは、集計値を算出するステップ８０６の後に、グローバル項目値番号の組から項目値の組を復元し、項目値の組、及び、項目値の組に対応した集計値を含む結果テーブルを生成することができる（ステップ８０７）。この結果テーブル自体は、テーブルの形で保持されているので、このテーブルをさらに集計することにより、別の次元に関する集計を簡単に得ることができる。例えば、性別・年齢別で得られた集計結果から性別の集計結果を簡単に作成することができる。

【００７０】

図５に示した表形式データに基づいて、実施例１による集計処理をより詳細に説明する。例えば、図５に示されるような表形式データを複数の処理モジュールで分掌管理し、上

述のステップ８０１を実行すると、図７に示されるようなデータの記憶構造が得られる。

【００７１】

ここで、図７に示されたようなデータに対して、「性別・年齢別に人数を求める」という集計処理や、「性別・年齢別に身長の合計値を求める」という集計処理を適用することができる。ここで、性別及び年齢はディメンジョン（次元）であり、人数や身長はメジャー（測度）である。人数は、マッチングする次元に対応するカウントを指定数だけ増加させることにより集計することができ、身長の合計値は、マッチングする次元に対応した身長の項目値を加算することにより集計することができる。実施例１では、この「性別・年齢別に身長の合計値を求める」個数付きカウント処理について説明する。

【００７２】

ステップ８０２では、図７に示されたデータに対して、各処理モジュール、即ち、各ローカル環境において、次元「年齢」と次元「性別」の２次元に関して、各ローカル環境のレコードをソートする。２次元以上でソートする場合には、適当な次元の順にソートを段階的に行うことになる。一般的には、項目値の種類の個数が多いほどソートに伴う順番の入れ替えが頻繁に生じるので、項目値の種類の個数の多い次元から順番にソートする方が効率的である。本例では、性別と年齢を比べると、性別の項目値の種類（男と女の２個）よりも年齢の項目値の種類（１歳、２歳及び３歳の３個）の方が多いので、年齢、性別の順にソートを行う。

【００７３】

本実施の形態では、ローカル環境下でのソートは、並び順番号の配列OrdSetの要素の順番を入れ替えることに相当する。例えば、初期の並び順番号の配列の要素が０（即ち、レコード０）、１（即ち、レコード１）、２（即ち、レコード２）であり、レコード０の年齢の値が３歳、レコード１の年齢の値が１歳、レコード２の年齢の値が２歳である場合を考える。年齢の若い順にレコードを並べ替えると、レコード１、レコード２、レコード３の順番になる。このときのソート結果は、並び順番号の配列を１、２、０の順番に入れ替えることによって表される。並び順番号の配列OrdSetには、後の処理のため、このローカル・ソート後の順番を設定しておく。

【００７４】

図９は、図７で示されたデータに対して、各処理モジュール内で年齢によるソートと性別によるソートを順次適用した結果を示す図である。尚、図９では、簡単のため、身長に関する情報ブロックは示されていない。年齢によるソートは年齢の若い順に行われ、性別によるソートは、男性、女性の順に行われる。このローカル・ソートによって、ＰＭＭ－０内のレコードは、レコード１（男性、１歳、８２ｃｍ）、レコード２（女性、２歳、６９ｃｍ）、レコード０（女性、３歳、７８ｃｍ）の順に順序付けされ、ＰＭＭ－１内のレコードは、レコード１（男性、３歳、９１ｃｍ）、レコード０（女性、１歳、８２ｃｍ）の順に順序付けされ、ＰＭＭ－２内のレコードは、レコード０（女性、１歳、７６ｃｍ）、レコード１（女性、１歳、７８ｃｍ）、レコード２（女性、２歳、８４ｃｍ）の順に順序付けされ、ＰＭＭ－３内のレコードは、レコード０（男性、３歳、８７ｃｍ）、レコード１（女性、３歳、８０ｃｍ）の順に順序付けされる。このローカル・ソートについては、後述する。

【００７５】

ステップ８０３では、各処理モジュールは、ローカル・ソートされたレコード順（即ち、入れ替え後の並び順番号の配列OrdSetの要素の順）に、選択された次元（本例では、性別及び年齢）の項目値番号の組に順序番号を付ける。図１０は、本例における順序番号付与の説明図である。簡単のため、身長に関する情報ブロックは省略されている。

【００７６】

ＰＭＭ－０のレコード１に関して説明すると、性別の値番号は、「０」であり、値番号「０」に対応したグローバル項目値番号は「０」である。また、年齢の値番号は、「０」であり、年齢の値番号「０」に対応したグローバル項目値番号は「０」である。よって、ＰＭＭ－０内のレコード１に対応した項目値番号の組は、（０，０）であり、この（０，

０）にローカル・ソートされたレコード順に順序番号が付けられる。項目値番号の組に順序番号を付与することは、内部的には、性別のグローバル値項目値番号と年齢のグローバル項目値番号に同一の順序番号「０」を与えることによって実現できる。本例では、ＰＭＭ－０内のレコード１はローカル・ソートされた１番目のレコードであるため、対応した項目値番号の組（０，０）に対して、順序番号「０」が付与される。ＰＭＭ－０内のレコードは、レコード１、レコード２、レコード０の順に順序付けられているので、以下、レコード２に対応したグローバル項目値番号の組（１，１）に順序番号「１」を与え、レコード０に対応したグローバル項目値番号の組（１，２）に順序番号「２」を与える。

【００７７】

順序番号は、レコードに対応して設定されるので、ＰＭＭ内に次元値が一致するレコードが２個以上存在する場合には、レコード毎に別々の順序番号が付与されている。例えば、図１０において、ＰＭＭ－２内のレコード０とレコード１は、共に、性別が女性であり、年齢が１歳であるため、グローバル項目値番号の組は、どちらも（０，１）である。検索やソートが目的である場合には、グローバル値番号の組が一致していても、異なるレコードは別個に取り扱う必要がある。例えば、グローバル項目値番号の組と、グローバル・レコード番号GOrdとを組み合わせて使用することによって、すべてのレコードを別個に取り扱うことができる。しかし、本例のように集計を目的とする場合、グローバル項目値番号の組が一致するレコード、即ち、次元値が一致するレコードは、同じ次元として取り扱う方が都合がよい。そのため、本実施の形態では、グローバル項目値番号の組が一致するレコードには、同じ番号が付与されるように順序番号を付与し直す。このように付け直された順序番号を、以下では、ローカル次元値番号LDimNoと称する。ローカル次元値番号は、グローバル項目値番号の組が異なる場合に初めて１ずつインクリメントされるような番号である。図１１は、このようなローカル次元値番号付与の説明図である。本例では、ＰＭＭ－０、ＰＭＭ－１及びＰＭＭ－３内では、順序番号とローカル次元値番号は同じであるが、ＰＭＭ－２内では、ローカル次元値番号は、ソートされたレコードの順に、「０」、「０」、「１」のようになる。

【００７８】

次に、ステップ８０４において、各処理モジュールは、グローバル項目値番号の組に対して付与されたローカル次元値番号LDimNoを、複数の処理モジュール間で共通のグローバル次元値番号GDimNoに変換することにより、次元値に対してグローバルな順位付けを行う。次元値にグローバルな順位付けがなされると、後述のように、各処理モジュール内で次元値毎に集計を行った後、その集計結果を統合することにより、全体の集計結果を得ることができるようになる。

【００７９】

図１２はグローバル次元値番号付与の説明図である。このグローバル次元値番号付与は、各処理モジュール内で順位付けされたグローバル項目値番号の組を、複数の処理モジュール間で共通に順位付けすることである。そのため、各処理モジュールは、グローバル次元値番号GDimNoの領域を確保し、ローカル次元値番号LDimNoからグローバル次元値番号GDimNoの初期値を生成する。同じローカル次元値番号LDimNoが割り当てられているレコードに対しては、一つのグローバル次元値番号GDimPosの領域だけを確保する。そのため、グローバル次元値番号GDimNoからローカル次元値番号LDimNoへの対応表GDimPosも同時に作成する。

【００８０】

次に、他の処理モジュールからグローバル項目値番号の組を相互に取得し、自処理モジュール内のグローバル項目値番号の組よりも前に順序付けされる組の個数をカウントし、自処理モジュール内のグローバル項目値番号の組のグローバル次元値番号をカウントされた個数分だけ引き上げることにより、グローバル項目値番号の組に対して、複数の処理モジュール間で共通のグローバル次元値番号を付ける。このように、各処理モジュールで個別に順序付けされている値、即ち、ローカル次元値番号に、複数の処理モジュール間で共通の順序番号、即ち、グローバル次元値番号を付与する方法は、同一の値に別個のグロー

バル次元値番号が付与されないように、同一の値の重複を排除する必要がある。この順序番号付与方法については後述する。

【００８１】

図１３は、図１２の例による処理モジュールＰＭＭ－０からＰＭＭ－３内における、ローカル次元値番号LDimNo、性別のグローバル項目値番号GVNo1、年齢のグローバル項目値番号GVNo2、及び、グローバル次元値番号GDimNoをまとめた図表である。同図からわかるように、グローバル次元値番号GDimNoは、性別のグローバル項目値番号GVNo1を上位の桁とし、年齢のグローバル項目値番号GVNo2を下位の桁としたときのグローバル項目値番号の組の番号順に０，１，２，４の番号が割り当てられている。

【００８２】

ステップ８０５において、各処理モジュールは、自処理モジュール内で、グローバル項目値番号の組毎に、即ち、グローバル次元値番号毎に項目値を集計する。本例では、ＰＭＭ－０からＰＭＭ－３は、各モジュール内で性別・年齢別に身長の値を合計する。

【００８３】

図１４及び１５は、各処理モジュール内でグローバル次元値番号毎に項目値を集計する処理の説明図である。最初に、図１４に示されるように、メジャーを格納するための領域として、グローバル次元値番号GDimNoの配列と同じサイズの配列GMsrを作成する。本例では、身長の合計を集計するので、浮動小数点又は整数のような格納領域を作成する。次に、図１５に示されるように、各処理モジュール内で、例えば、グローバル次元値番号の組の順に入れ替えられた配列OrdSetの要素の順に、集計すべき項目値を取り出し、メジャー配列GMsrに集計する。

【００８４】

例えば、ＰＭＭ－０では、並び順番号の配列OrdSetの先頭の要素が「１」番（即ち、レコード１）であることが分かるので、身長の情報ブロック内の値リストへのポインタ配列VNoのインデックス「１」の内容を参照する。ポインタ配列には、値「２」が格納されているので、ＰＭＭ－０のレコード０に関する身長の値は、値リストVLのインデックス「２」の内容を獲得することにより、「８２」として得られる。この値「８２」がメジャー配列GMsrに集計されるべき値である。本例では、集計値は合計値であるので、この値「８２」が加算される。

【００８５】

次に、この値「８２」を、メジャー配列GMsrのどの要素に加算すべきであるか、即ち、メジャー配列GMsrのインデックスを特定する必要がある。上述のように、ローカル次元値番号LDimNoの配列は、グローバル次元値番号の組の順に並べられているので、並び順番号の配列OrdSetとローカル次元値番号の配列LDimNoの要素の並び順、即ち、インデックスは対応している。そのため、配列OrdSetの先頭に関するメジャーは、配列LDimNoの先頭によって示されるメジャー配列GMsrの格納領域に集計すればよい。図１５の例では、配列OrdSetの先頭に対応した配列LDimNoの先頭の要素は「０」であるので、値「８２」は、メジャー配列GMsrのインデックス「０」で示される場所に加算される。

【００８６】

ＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３についても同様に配列OrdSetの先頭の要素について身長の値「９１」、「７６」及び「８７」を取得し、メジャー配列GMsrの先頭の領域に集計する。以下、ＰＭＭ－０からＰＭＭ－３に関して、配列OrdSetの２番目以降の要素についても同様に項目値を取得してメジャー配列GMsrに集計する。

【００８７】

尚、ＰＭＭ－２では、配列OrdSetの先頭の要素「０」と２番目の要素「１」に対して、ローカル次元値番号LDimNoが共に「０」であるため、対応した身長の値「７６」と「７８」は、共に、メジャー配列GMsrの先頭の領域に集計されるので、配列GMsrの先頭の集計結果は、７６＋７８＝１５４になる。

【００８８】

続いて、ステップ８０６では、各処理モジュールは、他の処理モジュールからグローバ

ル項目値番号の組毎のローカル集計値を取得し、取得された集計値をグローバル項目値番号の組毎に集計することにより集計値を算出する。このグローバルな集計は、処理モジュール間の物理的な伝送路の構成に応じて、例えば、次の２通りの方法で実現できる。

【００８９】

第１のグローバル集計方法では、各処理モジュールは、グローバル次元値番号GDimNoと、グローバル次元値番号GDimNoに対応して集計されたメジャーGMsrの組を他の処理モジュールへ送信する。この方法は、処理モジュール間に複数の伝送路を確保できる場合に適している。図１６は、第１のグローバル集計方法の説明図である。ＰＭＭ－０（符号１６００）、ＰＭＭ－１（符号１６０１）、ＰＭＭ－２（符号１６０２）及びＰＭＭ－３（符号１６０３）の４台の処理モジュールは、全体として１６０４で表される伝送路を介して接続されている。

【００９０】

例えば、処理モジュールＰＭＭ－０は、図１５で示されるようなローカル環境下での集計結果として、グローバル次元値番号GDimNoとメジャーGSMsrの３個の組、即ち、

（０，８２）

（３，６９）

（４，７８）

を他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３へ送信する。また、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－１から送信された２個の組、

（１，９１）

（２，８２）

と、処理モジュールＰＭＭ－２から送信された２個の組、

（２，１５４）

（３，８４）

と、処理モジュールＰＭＭ－３から送信された２個の組、

（１，８７）

（４，８０）

を伝送路１６０４を介して受け取る。処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３についても同様に自処理モジュールのローカル集計結果を他のモジュールへ送信すると共に、他の処理モジュールからローカル集計結果を受信する。

【００９１】

各処理モジュールは、相互に交換したローカル集計結果を、各処理モジュール内でグローバル次元値番号毎に加算して、グローバル集計結果を算出する。図１７は、グローバル集計結果算出の説明図である。各処理モジュールが他の処理モジュールから受信したローカル集計結果の中で、実際に各処理モジュール内でグローバル集計に利用されるデータは、各処理モジュールのローカル集計結果のグローバル次元値番号と一致するグローバル次元値番号を含むデータだけである。図１７では、各処理モジュールが他の処理モジュールから受信したデータの中で、実際のグローバル集計に利用されないデータが二重取消線で示されている。そして、各処理モジュールは、他の処理モジュールから受信したメジャーを並列的に加算することも可能である。これにより、全体的な処理速度が高速化できる。

【００９２】

ローカル集計結果の加算により、図１７に示されるように、各処理モジュールには、グローバル集計結果が得られる。例えば、グローバル次元値番号「０」は、元々、ＰＭＭ－０だけに存在していたので、グローバル次元値番号「０」についての集計結果は、ＰＭＭ－０だけに現れる。一方、グローバル次元値番号「３」は、ＰＭＭ－０とＰＭＭ－２の２個の処理モジュールにおいてローカル集計されていたので、グローバル次元値番号「３」に対応するグローバル集計結果は、ＰＭＭ－０とＰＭＭ－２の２個の処理モジュールに現れている。もちろん、グローバル次元値番号「３」に関するＰＭＭ－０とＰＭＭ－２の両方のグローバル集計値は、同じ値「１５３」をとる。

【００９３】

このような重複したグローバル集計結果は削除しておいた方が後の処理のために都合がよい。そのため、処理モジュールに予め順位を割り当てておき、各処理モジュールは、自処理モジュールよりも上位の処理モジュールが自処理モジュールで保持しているグローバル次元値番号についてのグローバル集計値と同じグローバル集計値を保持している場合には、自処理モジュールで保持しているグローバル集計値を削除するように構成することができる。図１８は、このようなグローバル集計値の重複を排除するための処理の説明図である。同図において、二重取消線は、グローバル集計値の重複の排除を表している。この処理を加えることにより、すべての処理モジュールを通じて、各グローバル次元値番号に対して１個のグローバル集計値が保持されることになる。

【００９４】

最後に、ステップ８０７では、最終的なグローバル集計値を保持している処理モジュールは、グローバル項目値番号の組から項目値の組を復元し、項目値の組、及び、項目値の組に対応した集計値を含む結果テーブルを生成する。図１９は、結果テーブル生成の説明図である。集計結果をこのような結果テーブルの形で表現することにより、更なる集計処理に利用できるという利点が得られる。図１８の例では、最終的なグローバル集計値は、処理モジュールＰＭＭ－０とＰＭＭ－１に保持されているので、処理モジュールＰＭＭ－０とＰＭＭ－１において、結果テーブルを作成すればよい。

【００９５】

例えば、処理モジュールＰＭＭ－０において、グローバル次元値番号「０」のグローバル集計結果は「８２」である。グローバル次元値番号GDimNo「０」に対するローカル次元値番号LDimNoは、図１２を参照して説明したグローバル次元値番号からローカル次元値番号への対応表GDimPosを用いることにより獲得することができる。図１９の例では、GDimNo「０」に対するGDimPosの値は「０」であるため、配列LDimNoの先頭の要素「０」がローカル次元値番号である。ローカル次元値番号「０」には、性別のグローバル項目値番号「０」と年齢のグローバル項目値番号「０」が対応している。性別のグローバル項目値番号「０」に対応する項目値、即ち、次元値は「男」であり、年齢のグローバル項目値番号「０」に対応する項目値、即ち、次元値は「１」である。よって、グローバル次元値番号「０」に対して、性別の次元値「男」と、年齢の次元値「１」と、集計値（＝身長の合計値）「８２」を得ることができる。処理モジュールＰＭＭ－０の他のグローバル次元値番号と、処理モジュールＰＭＭ－１のグローバル次元値番号に対しても同様の処理を適用することにより、結果テーブルを得ることができる。図２０は、このようにして作成された結果テーブルの説明図である。処理モジュールＰＭＭ－０とＰＭＭ－１は、性別次元値・年齢次元値・集計値の結果テーブルを生成する。処理モジュールＰＭＭ－２とＰＭＭ－３は、結果テーブルを生成しない。

【００９６】

図１６乃至１８では、第１のグローバル集計方法について説明したが、本実施の形態の変形例では、第２のグローバル集計方法が実施される。図２１は、第２のグローバル集計方法の説明図である。この集計方法は、処理モジュールに予め順位を割り当てておき、ローカル集計結果である配列GMsrを上位の処理モジュールから下位の処理モジュールへ順次送る。２番目以降の処理モジュールでは、前の処理モジュールから受信した集計結果配列GMsrに自処理モジュール内でのローカル集計結果を合算し、合算後の集計結果配列GMsrを次の処理モジュールへ送信する。このように、集計結果を合算しながら集計結果配列GMsrを後続の処理モジュールへ順送りすることにより、処理モジュールの連なりを一巡して最初の最上位の処理モジュールへ戻される集計結果配列は、すべてのグローバル次元値番号に関するグローバル集計結果を格納した配列である。

【００９７】

図２１の例では、最初に、最上位の処理モジュールＰＭＭ－０から次の処理モジュールＰＭＭ－１へ集計結果配列（８２，－，－，６９，７８）が送信される。ここで、「－」は、ローカル集計結果が存在しないことを表す。処理モジュールＰＭＭ－１は、受信した集計結果配列（８２，－，－，６９，７８）に自処理モジュール内のローカル集計結果（

－，９１，８２，－，－）を合わせて更なる集計結果配列（８２，９１，８２，６９，７８）を生成し、次の処理モジュールＰＭＭ－２へ送信する。ＰＭＭ－２も同様に、受信した集計結果配列に、自処理モジュール内のローカル集計結果（－，－，１５４，８４，－）を合わせて更なる集計結果配列（８２，９１，２３６，１５３，７８）を生成し、次の処理モジュールＰＭＭ－３へ送信する。ＰＭＭ－３も同様に、受信した集計結果配列に、自処理モジュール内のローカル集計結果（－，８７，－，－，８０）を合わせて、更なる集計結果配列（８２，１７８，２３６，１５３，１５８）を生成する。ＰＭＭ－３は、最下位の処理モジュールであるため、ＰＭＭ－３から出力される集計結果配列は、最終的な集計結果である。

【００９８】

[順序番号付与処理]

本実施の形態による情報処理システムのように、順序付きの値のリストを格納しているメモリをそれぞれに有する複数の処理モジュールが論理的に環状に接続されている情報処理システムでは、各処理モジュールで個別に順序付けされている値に複数の処理モジュール間で共通の順序番号を付与する情報処理方法、即ち、順序番号付与方法が必要である。

【００９９】

例えば、図１２を参照して説明したように、グローバル次元値番号を付与する際には、各処理モジュールで個別に順序付けされている値に、複数の処理モジュール間で共通の順序番号を付与する順序番号付与処理が使用される。また、この順序番号付与処理は、グローバル次元値番号を付与する場合の他に、後述のコンパイル処理において、グローバル項目値番号を設定する場合にも使用される。この順序番号付与処理は、同一値には唯一の番号しか付与しないことを特徴としている。したがって、このタイプの順序番号付与処理は、特に、同一値消去型の順序番号付与処理と称する。

【０１００】

図２２は、実施例１における順序番号付与方法のフローチャートである。同図に示されるように、各処理モジュールは、自処理モジュール内の値のリスト中の各値の順序番号の初期値をメモリに格納する（ステップ２２０１）。

【０１０１】

次に、各処理モジュールは、自処理モジュールのメモリに格納されている値のリストを論理的に次の段に接続された処理モジュールへ送信する（ステップ２２０２）。更に、各処理モジュールは、自処理モジュール内の値のリスト中の各値に対し、論理的に前の段に接続された処理モジュールから受信した値のリストの中から上記各値よりも前に順序付けされる値の個数をカウントし、自処理モジュール内の値のリスト中の各値の順序番号をカウントされた個数分だけ引き上げることにより、自処理モジュール内の値のリスト中の各値の順序番号を更新し、更新された順序番号をメモリに格納する（ステップ２２０３）。

【０１０２】

次に、各処理モジュールは、受信した値のリスト中の値から自処理モジュール内の値のリスト中の値と一致する値を除いた更なる値のリストを論理的に次の段に接続された処理モジュールへ送信し（ステップ２２０４）、各処理モジュールは、自処理モジュール内の値のリスト中の各値に対し、論理的に前の段に接続された処理モジュールから受信した更なる値のリストの中から上記各値よりも前に順序付けされる値の個数をカウントし、自処理モジュール内の値のリスト中の各値の順序番号をカウントされた個数分だけ引き上げることにより、自処理モジュール内の値のリスト中の各値の順序番号を更新し、更新された順序番号をメモリに格納する（ステップ２２０５）。

【０１０３】

続いて、各処理モジュールは、送信ステップ２２０２において論理的に次の段に接続された処理モジュールへ送信した値のリストが、論理的に環状に接続された他の処理モジュールを介して、論理的に前の段に接続された処理モジュールによって受信されるまで、ステップ２２０４とステップ２２０５を繰り返し実行する（ステップ２２０６）。

【０１０４】

この順序番号付与方法によれば、各処理モジュールは、他の処理モジュールが保持する値のリストを重複無く受け取り、自処理モジュールが保持する値にグローバルな順序番号を付与することができる。上述のように、各処理モジュールが予め順序付きの値のリストを保持している場合には、非常に効率的にグローバルな順序番号を付与することが可能である。なぜならば、値のリストが予め順序付けされている場合には、昇順（又は降順）の一方向に順序を比較するだけでよいからである。もちろん、各処理モジュールの保持する値のリストが順序付きでない場合でも同様の結果を得ることができる。その場合には、例えば、各処理モジュールは、他の処理モジュールから受信した値のリスト中の値と、自処理モジュールが保持している値のリスト中の値を、すべての組合せについて順番に比較して、各値よりも前に、即ち、上位に、順序付けされる値の個数をカウントして、各値の順序番号を更新すればよい。

【０１０５】

本実施の形態の順序番号付与方法では、各処理モジュールは、他の処理モジュールから受信した値のリストを保存する必要はなく、自処理モジュールが保持している値のリストに順序を付けるだけで、すべての処理モジュールに共通の順序番号を付与することができる。また、この順序番号付与方法は、他の処理モジュールからの値のリストの受信順に影響されないので、処理モジュール間の物理的な接続形態に全く依存しない。したがって、伝送路と順序番号更新回路を多重化することによって、更なる高速化を実現することができる。

【０１０６】

図２３及び図２４は、順序番号付与処理の説明図である。図２３は、各ＰＭＭが次の段に接続されたＰＭＭへ送信する値のリストがステップ毎に示されている。図２４は、各ステップで、ＰＭＭが前の段に接続されたＰＭＭから受信した値のリストが示されている。この例では、初期状態として、ＰＭＭ－０が値のリスト［１８，２１，２４］を保持し、ＰＭＭ－１が値のリスト［１６，２８］を保持し、ＰＭＭ－２が値のリスト［１６，２０，３３］を保持し、ＰＭＭ－３が値のリスト［１８，２４］を保持している。

【０１０７】

ステップ３の終了時点で、各ＰＭＭは、他のすべての処理モジュールからの値のリストを受信することができる。この時点で、自処理モジュールが保持している値のリストと、受信した値のリストを合わせることにより、すべての値の順序を決めることができる。更に、ステップ４の終了時点では、すべての値を重複無く受信できることが分かる。

【０１０８】

［コンパイル処理］

コンパイル処理は、各処理モジュールでデータを管理するために使用するグローバル・レコード番号GOrdとグローバル項目値番号GVNoを設定するための処理である。グローバル・レコード番号GOrdは、上述のオフセット値OFFSETを使用することにより簡単に設定することができる。一方、グローバル項目値番号GVNoは、各処理モジュールが個別に保持している値リストに基づいて、すべての処理モジュール間で共通に順序付けされる番号である。

【０１０９】

したがって、各処理モジュールは、上述の順序番号付与処理を用いることによってグローバル項目値番号GVNoを設定することが可能である。

【実施例２】

【０１１０】

実施例２では、図５に示した表形式データに基づいて、「性別・年齢別に人数を求める」個数付きカウント処理を行う集計を説明する。実施例２においても、実施例１と同様に、図８に示したフローチャートに沿って集計処理が行われる。

【０１１１】

例えば、図５に示されるような表形式データを複数の処理モジュールで分掌管理し、上述のステップ８０１を実行すると、図７に示されるようなデータの記憶構造が得られる。

ステップ８０２では、図７に示されたデータに対して、各処理モジュール、即ち、各ローカル環境において、次元「年齢」と次元「性別」の２次元に関して、各ローカル環境のレコードをソートする。実施例２におけるステップ８０２は、実施例１におけるステップ1と同様である。

【０１１２】

ステップ８０３では、各処理モジュールは、ローカル・ソートされたレコード順（即ち、入れ替え後の並び順番号の配列OrdSetの要素の順）に、選択された次元（本例では、性別及び年齢）の項目値番号の組に順序番号を付ける。実施例２におけるステップ８０３は、実施例１におけるステップ８０３と同様である。

【０１１３】

次に、ステップ８０４において、各処理モジュールは、グローバル項目値番号の組に対して付与されたローカル次元値番号LDimNoを、複数の処理モジュール間で共通のグローバル次元値番号GDimNoに変換することにより、次元値に対してグローバルな順位付けを行う。実施例２におけるステップ８０４は、実施例１におけるステップ８０４と同様である。

【０１１４】

続いて、ステップ８０５において、各処理モジュールは、自処理モジュール内で、グローバル項目値番号の組毎に、即ち、グローバル次元値番号毎に項目値を集計する。本例では、ＰＭＭ－０からＰＭＭ－３は、各モジュール内で性別・年齢別に人数をカウントする。

【０１１５】

図２５及び２６は、各処理モジュール内でグローバル次元値番号毎に項目値を集計する処理の説明図である。最初に、図２５に示されるように、メジャーを格納するための領域として、グローバル次元値番号GDimNoの配列と同じサイズの配列GMsrを作成する。本例では、人数を集計するので、整数の格納領域を作成する。次に、図２６に示されるように、各処理モジュール内で、例えば、グローバル次元値番号の組の順に入れ替えられた配列OrdSetの要素の順に、集計すべき値を取り出し、メジャー配列GMsrに集計する。

【０１１６】

本例では、集計すべき値は、「１」である。次に、この値「１」を、メジャー配列GMsrのどの要素に加算すべきであるか、即ち、メジャー配列GMsrのインデックスを特定する必要がある。上述のように、ローカル次元値番号LDimNoの配列は、グローバル次元値番号の組の順に並べられているので、並び順番号の配列OrdSetとローカル次元値番号の配列LDimNoの要素の並び順、即ち、インデックスは対応している。そのため、配列OrdSetの先頭に関するメジャーは、配列LDimNoの先頭によって示されるメジャー配列GMsrの格納領域に集計すればよい。図２６の例では、配列OrdSetの先頭に対応した配列LDimNoの先頭の要素は「０」であるので、値「１」は、メジャー配列GMsrのインデックス「０」で示される場所に加算される（インクリメントされる）。

【０１１７】

ＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３についても同様に、メジャー配列GMsrの先頭の領域をインクリメントする。以下、ＰＭＭ－０からＰＭＭ－３に関して、配列OrdSetの２番目以降の要素についても同様にメジャー配列GMsrをインクリメントする。

【０１１８】

尚、ＰＭＭ－２では、配列OrdSetの先頭の要素「０」と２番目の要素「１」に対して、ローカル次元値番号LDimNoが共に「０」であるため、対応した人数は、共に、メジャー配列GMsrの先頭の領域に集計されるので、配列GMsrの先頭の集計結果は、１＋１＝２になる。

【０１１９】

続いて、ステップ８０６では、各処理モジュールは、他の処理モジュールからグローバル項目値番号の組毎のローカル集計値を取得し、取得された集計値をグローバル項目値番号の組毎に集計することにより集計値を算出する。本例では、人数を指定数だけ増やすので個数付きカウント処理を行うことになる。このグローバルな集計は、実施例１と同様に

、処理モジュール間の物理的な伝送路の構成に応じて、例えば、次の２通りの方法で実現できる。

【０１２０】

第１のグローバル集計方法では、各処理モジュールは、グローバル次元値番号GDimNoと、グローバル次元値番号GDimNoに対応して集計されたメジャーGMsrの組を他の処理モジュールへ送信する。この方法は、処理モジュール間に複数の伝送路を確保できる場合に適している。図２７は、第１のグローバル集計方法の説明図である。ＰＭＭ－０（符号２７００）、ＰＭＭ－１（符号２７０１）、ＰＭＭ－２（符号２７０２）及びＰＭＭ－３（符号２７０３）の４台の処理モジュールは、全体として２７０４で表される伝送路を介して接続されている。

【０１２１】

例えば、処理モジュールＰＭＭ－０は、図２６で示されるようなローカル環境下での集計結果として、グローバル次元値番号GDimNoとメジャーGSMsrの３個の組、即ち、
（０，１）
（３，１）
（４，１）
を他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３へ送信する。また、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－１から送信された２個の組、
（１，１）
（２，１）
と、処理モジュールＰＭＭ－２から送信された２個の組、
（２，２）
（３，１）
と、処理モジュールＰＭＭ－３から送信された２個の組、
（１，１）
（４，１）
を伝送路２７０４を介して受け取る。処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３についても同様に自処理モジュールのローカル集計結果を他のモジュールへ送信すると共に、他の処理モジュールからローカル集計結果を受信する。

【０１２２】

各処理モジュールは、相互に交換したローカル集計結果を、各処理モジュール内でグローバル次元値番号毎に加算して、グローバル集計結果を算出する。図２８は、グローバル集計結果算出の説明図である。各処理モジュールが他の処理モジュールから受信したローカル集計結果の中で、実際に各処理モジュール内でグローバル集計に利用されるデータは、各処理モジュールのローカル集計結果のグローバル次元値番号と一致するグローバル次元値番号を含むデータだけである。図２８では、各処理モジュールが他の処理モジュールから受信したデータの中で、実際のグローバル集計に利用されないデータが二重取消線で示されている。そして、各処理モジュールは、他の処理モジュールから受信したメジャーを並列的に加算することも可能である。これにより、全体的な処理速度が高速化できる。

【０１２３】

ローカル集計結果の加算により、図２８に示されるように、各処理モジュールには、グローバル集計結果が得られる。例えば、グローバル次元値番号「０」は、元々、ＰＭＭ－０だけに存在していたので、グローバル次元値番号「０」についての集計結果は、ＰＭＭ－０だけに現れる。一方、グローバル次元値番号「３」は、ＰＭＭ－０とＰＭＭ－２の２個の処理モジュールにおいてローカル集計されていたので、グローバル次元値番号「３」に対応するグローバル集計結果は、ＰＭＭ－０とＰＭＭ－２の２個の処理モジュールに現れている。もちろん、グローバル次元値番号「３」に関するＰＭＭ－０とＰＭＭ－２の両方のグローバル集計値は、同じ値「２」をとる。

【０１２４】

このような重複したグローバル集計結果は削除しておいた方が後の処理のために都合が

よい。そのため、処理モジュールに予め順位を割り当てておき、各処理モジュールは、自処理モジュールよりも上位の処理モジュールが自処理モジュールで保持しているグローバル次元値番号についてのグローバル集計値と同じグローバル集計値を保持している場合には、自処理モジュールで保持しているグローバル集計値を削除するように構成することができる。図２９は、このようなグローバル集計値の重複を排除するための処理の説明図である。同図において、二重取消線は、グローバル集計値の重複の排除を表している。この処理を加えることにより、すべての処理モジュールを通じて、各グローバル次元値番号に対して１個のグローバル集計値が保持されることになる。

【０１２５】

最後に、ステップ８０７では、最終的なグローバル集計値を保持している処理モジュールは、グローバル項目値番号の組から項目値の組を復元し、項目値の組、及び、項目値の組に対応した集計値を含む結果テーブルを生成する。図３０は、結果テーブル生成の説明図である。集計結果をこのような結果テーブルの形で表現することにより、更なる集計処理に利用できるという利点が得られる。図２９の例では、最終的なグローバル集計値は、処理モジュールＰＭＭ－０とＰＭＭ－１に保持されているので、処理モジュールＰＭＭ－０とＰＭＭ－１において、結果テーブルを作成すればよい。

【０１２６】

例えば、処理モジュールＰＭＭ－０において、グローバル次元値番号「０」のグローバル集計結果は「１」である。グローバル次元値番号GDimNo「０」に対するローカル次元値番号LDimNoは、図１２を参照して説明したグローバル次元値番号からローカル次元値番号への対応表GDimPosを用いることにより獲得することができる。図３０の例では、GDimNo「０」に対するGDimPosの値は「０」であるため、配列LDimNoの先頭の要素「０」がローカル次元値番号である。ローカル次元値番号「０」には、性別のグローバル項目値番号「０」と年齢のグローバル項目値番号「０」が対応している。性別のグローバル項目値番号「０」に対応する項目値、即ち、次元値は「男」であり、年齢のグローバル項目値番号「０」に対応する項目値、即ち、次元値は「１」である。よって、グローバル次元値番号「０」に対して、性別の次元値「男」と、年齢の次元値「１」と、集計値（＝人数）「１」を得ることができる。処理モジュールＰＭＭ－０の他のグローバル次元値番号と、処理モジュールＰＭＭ－１のグローバル次元値番号に対しても同様の処理を適用することにより、結果テーブルを得ることができる。図３１は、このようにして作成された結果テーブルの説明図である。処理モジュールＰＭＭ－０とＰＭＭ－１は、性別次元値・年齢次元値・集計値の結果テーブルを生成する。処理モジュールＰＭＭ－２とＰＭＭ－３は、結果テーブルを生成しない。

【０１２７】

図２７乃至２９では、第１のグローバル集計方法について説明したが、本実施の形態の変形例では、第２のグローバル集計方法が実施される。図３２は、第２のグローバル集計方法の説明図である。この集計方法は、処理モジュールに予め順位を割り当てておき、ローカル集計結果である配列GMsrを上位の処理モジュールから下位の処理モジュールへ順次送る。２番目以降の処理モジュールでは、前の処理モジュールから受信した集計結果配列GMsrに自処理モジュール内でのローカル集計結果を合算し、合算後の集計結果配列GMsrを次の処理モジュールへ送信する。このように、集計結果を合算しながら集計結果配列GMsrを後続の処理モジュールへ順送りすることにより、処理モジュールの連なりを一巡して最初の最上位の処理モジュールへ戻される集計結果配列は、すべてのグローバル次元値番号に関するグローバル集計結果を格納した配列である。

【０１２８】

図３２の例では、最初に、最上位の処理モジュールＰＭＭ－０から次の処理モジュールＰＭＭ－１へ集計結果配列（１，－，－，１，１）が送信される。ここで、「－」は、ローカル集計結果が存在しないことを表す。処理モジュールＰＭＭ－１は、受信した集計結果配列（１，－，－，１，１）に自処理モジュール内のローカル集計結果（－，１，１，－，－）を合わせて更なる集計結果配列（１，１，１，１，１）を生成し、次の処理モジ

ュールＰＭＭ－２へ送信する。ＰＭＭ－２も同様に、受信した集計結果配列に、自処理モジュール内のローカル集計結果（－，－，２，１，－）を合わせて更なる集計結果配列（１，１，３，２，１）を生成し、次の処理モジュールＰＭＭ－３へ送信する。ＰＭＭ－３も同様に、受信した集計結果配列に、自処理モジュール内のローカル集計結果（－，１，－，－，１）を合わせて、更なる集計結果配列（１，２，３，２，２）を生成する。ＰＭＭ－３は、最下位の処理モジュールであるため、ＰＭＭ－３から出力される集計結果配列は、最終的な集計結果である。

【０１２９】

尚、実施例１で得られた集計値（＝身長の合計）を実施例２で得られた集計値（＝人数）で除算することにより、平均身長という集計値を得ることができる。

【実施例３】

【０１３０】

実施例１、２では、個数付きカウント処理を使用する集計処理について説明したが、マッチングした値の出現数を数えるカウント処理は、分散メモリ型の並列アーキテクチャに基づいて、プロセッサ間通信ができるだけ少なくなるように、大規模データをプロセッサ間で共有することなく、大規模データを個々のプロセッサ内に保持したまま、大規模データの処理を実現するために必要なプロセッサ間の最も基本的な情報処理技術である。

【０１３１】

このカウント処理を用いることにより、一つのモジュールには高々１個しか存在しない値が全体で何個存在しているかを知ることができる。

【０１３２】

例えば、ある組立工場がある製品Ａを製造するための多数の部品を多数の部品工場のうちのいずれかの部品工場から仕入れる場合に、部品工場毎に供給可能な部品の種類が異なることがある。このとき、ある部品について、その部品に不足が生じないようにするため、その部品を供給可能な部品工場の数を把握する際に、出現数を数えるカウント処理を利用することができる。即ち、各部品工場に各処理モジュールを対応させ、部品の型番を項目値に対応させると、指定した型番にマッチングする項目値の個数をカウントすることにより、その部品を供給可能な部品工場の数を獲得することができる。

【０１３３】

図３３は、各工場が供給可能な部品の型番を表す表形式データがＰＭＭ－０からＰＭＭ－３に分掌管理されている様子を示す図である。簡単のために、型番以外の項目（例えば、部品価格）などは図示されていない。また、型番のVNoに対応してカウント値を格納するための作業領域が設けられ、初期値として１が設定される。

【０１３４】

図３４は、各処理モジュールが、複数の伝送路を利用して、自処理モジュール内の項目値を他の処理モジュールへ送信することによって、マッチングする個数をカウントする処理の説明図である。

【０１３５】

ＰＭＭ－０（符号３４００）、ＰＭＭ－１（符号３４０１）、ＰＭＭ－２（符号３４０２）及びＰＭＭ－３（符号３４０３）の４台の処理モジュールは、全体として３４０４で表される伝送路を介して接続されている。例えば、処理モジュールＰＭＭ－０は、図３０で示されるようなローカル環境下での型番のグローバル項目値番号GVNoの組、即ち、

（０，１，２）

を他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３へ送信する。また、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－１、ＰＭＭ－２、及び、ＰＭＭ－３から送信されたグローバル項目値番号の組、即ち、

（１，３）

（０，２，３）

及び

（０，３）

を伝送路３４０４を介して受け取る。処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３についても同様に自処理モジュールの項目値番号の組を他のモジュールへ送信すると共に、他の処理モジュールから項目値番号の組を受信する。

【０１３６】

例えば、処理モジュールＰＭＭ－０のカウント値格納領域の初期値は、
GVNo「０」の格納領域：「１」
GVNo「１」の格納領域：「１」
GVNo「２」の格納領域：「１」
である。GVNo「３」のための格納領域は存在しない。

【０１３７】

ここで、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－１から（１，３）を受信すると、カウント値格納領域を、
GVNo「０」の格納領域：「１」
GVNo「１」の格納領域：「１」→「２」
GVNo「２」の格納領域：「１」
のようにカウントアップする。次に、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－２から（０，２，３）を受信すると、カウント値格納領域を、
GVNo「０」の格納領域：「１」→「２」
GVNo「１」の格納領域：「２」
GVNo「２」の格納領域：「１」→「２」
のようにカウントアップする。次に、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－３から（０，３）を受信すると、カウント値格納領域を、
GVNo「０」の格納領域：「２」→「３」
GVNo「１」の格納領域：「２」
GVNo「２」の格納領域：「２」
のようにカウントアップする。

【０１３８】

処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３についても、同様に、他の処理モジュールから項目値番号の組を受信し、自処理モジュール内に存在する項目値番号とマッチングする項目値に対応するカウントを１ずつインクリメントすると、
処理モジュールＰＭＭ－１のカウント値領域として、
GVNo「１」の格納領域：「２」
GVNo「３」の格納領域：「３」
処理モジュールＰＭＭ－２のカウント値領域として、
GVNo「０」の格納領域：「２」
GVNo「２」の格納領域：「２」
GVNo「３」の格納領域：「３」
処理モジュールＰＭＭ－３のカウント値領域として、
GVNo「０」の格納領域：「３」
GVNo「３」の格納領域：「３」
が得られる。

【０１３９】

この段階では、重複したカウント結果が得られているので、例えば、重複部分を削除することによって、最終的に、
GVNo「０」のカウント結果：「３」
GVNo「１」のカウント結果：「２」
GVNo「２」のカウント結果：「２」
GVNo「３」のカウント結果：「３」
が得られる。

【０１４０】

上記の例では、処理モジュールＰＭＭ－０は、順次カウントアップを行っているが、各処理モジュールから他の処理モジュールへの項目値の組の送受信と、各処理モジュール内でのカウントアップを並列的に実行することが可能である。例えば、GVNo「０」～GVNo「３」までのカウント値を、（GVNo「０」のカウント，GVNo「１」のカウント，GVNo「２」のカウント，GVNo「３」のカウント）のようにベクトル表記すると、処理モジュールＰＭＭ－０では、自モジュール内でのカウント値を表すベクトル（１，１，１，０）と、処理モジュールＰＭＭ－１から受信したカウント値を表すベクトル（０，１，０，１）と、処理モジュールＰＭＭ－２から受信したカウント値を表すベクトル（１，０，１，１）と、処理モジュールＰＭＭ－３から受信したカウント値を表すベクトル（１，０，０，１）を加算することによって、最終的なカウント値（３，２，２，３）を得ることができる。

【０１４１】

また、図３５は、処理モジュールに予め順位を割り当てておき、各処理モジュールのカウント値を、上位の処理モジュールから下位の処理モジュールへ順次送る。２番目以降の処理モジュールでは、前の処理モジュールから受信したカウント値に自処理モジュール内でのカウント値を合算し、合算後のカウント値を次の処理モジュールへ送信する。このように、カウント値を合算しながら合算後のカウント値を後続の処理モジュールへ順送りすることにより、処理モジュールの連なりを一巡して最初の最上位の処理モジュールへ戻されるカウント値は、最終的なカウント値（３，２，２，３）である。

【実施例４】

【０１４２】

実施例１から３では、マッチングする項目値に対応する個数をカウントしているが、大規模データ処理の分野では、データの順位付けが必要になることも多い。例えば、学力テストの科目毎の点数が表形式データとして得られている状況で、合計点を集計し、合計点の高い方から順位を付ける場合などが考えられる。

【０１４３】

本実施例でも、処理モジュールＰＭＭ０～ＰＭＭ３が多数の受験者のデータを分掌管理している場合を考える。ある受験者の成績は、同一の処理モジュールで管理されているならば、各受験者の合計点は、処理モジュール毎にローカルに算出することができる。そこで、本例では、各受験者の合計点が算出された後に合計点で順位付けをする処理について説明する。

【０１４４】

図３６は、本発明の実施例４による表形式データの記憶構造の一例の説明図である。同図に示されるように、新たに算出された合計点は、この表形式データの新しい項目として追加される。新しい項目「合計点」については、通常、ＰＭＭ間コンパイル処理によってグローバル項目値番号GVNoが割り当てられるが、本実施例では、グローバル項目値番号GVNoの他に、順位を表すランク(rank)が割り当てられる。グローバル項目値番号の付与は、値が同一の項目値に対しては同じ番号が割り当てられるという点で「同一値を消去する」タイプの順序付けであり、一方、ランクの付与は、値が同じ項目値であってもグローバル順序集合配列Gordの値が異なるので（本例では、別々の受験者であるので）、同一値を考慮してランク付けが行われる。

【０１４５】

グローバル項目値番号は、上述のコンパイル処理によって付与することができる。そこで、以下では、ランクを付与するための単純順序付け処理について説明する。

【０１４６】

図３７は、本発明の実施例４による処理モジュールＰＭＭ－０のランク及びグローバル項目値番号を付与する順序付け処理の説明図である。図３７の（ａ）は、処理モジュールＰＭＭ－０の初期状態を示し、同図の（ｂ）は処理モジュールＰＭＭ－１から項目値４４０，４１０を受信した後の状態を示し、同図の（ｃ）は処理モジュールＰＭＭ－２から項目値４２０，４１０，３８０を受信した後の状態を示し、同図の（ｄ）は処理モジュールＰＭＭ－３から項目値４５０，４４０を受信した後の状態を示している。

【０１４７】
グローバル項目値番号GVNoは、例えば、図２２～２４を参照して説明した順序番号付与方法に従って付与することができる。
【０１４８】
図３８は、実施例４におけるランク付与方法のフローチャートである。同図に示されるように、各処理モジュールは、自処理モジュール内の合計点のリスト中の各値のランクの初期値をメモリに格納する（ステップ３８０１）。
【０１４９】
次に、各処理モジュールは、自処理モジュールのメモリに格納されている値（本例では、合計点）のリストを論理的に次の段に接続された処理モジュールへ送信する（ステップ３８０２）。更に、各処理モジュールは、自処理モジュール内の値のリスト中の各値に対し、論理的に前の段に接続された処理モジュールから受信した値のリストの中から上記各値よりも前に順序付けされる値の個数をカウントし、自処理モジュール内の値のリスト中の各値のランクをカウントされた個数分だけ引き上げることにより、自処理モジュール内の値のリスト中の各値のランクを更新し、更新されたランクをメモリに格納する（ステップ３８０３）。
【０１５０】
次に、各処理モジュールは、受信した値のリスト中の値を論理的に次の段に接続された処理モジュールへ送信し（ステップ３８０４）、各処理モジュールは、自処理モジュール内の値のリスト中の各値に対し、論理的に前の段に接続された処理モジュールから受信した更なる値のリストの中から上記各値よりも前に順序付けされる値の個数をカウントし、自処理モジュール内の値のリスト中の各値のランクをカウントされた個数分だけ引き上げることにより、自処理モジュール内の値のリスト中の各値のランクを更新し、更新されたランクをメモリに格納する（ステップ３８０５）。
【０１５１】
続いて、各処理モジュールは、送信ステップ３８０２において論理的に次の段に接続された処理モジュールへ送信した値のリストが、論理的に環状に接続された他の処理モジュールを介して、論理的に前の段に接続された処理モジュールによって受信されるまで、ステップ３８０４とステップ３８０５を繰り返し実行する（ステップ３８０６）。
【０１５２】
このランク付与方法によれば、各処理モジュールは、他の処理モジュールが保持する値のリストを重複無く受け取り、自処理モジュールが保持する値にグローバルな順序番号を付与することができる。上述のように、各処理モジュールが予め順序付きの値のリストを保持している場合には、非常に効率的にグローバルなランクを付与することが可能である。なぜならば、値のリストが予め順序付けされている場合には、昇順（又は降順）の一方向に順序を比較するだけでよいからである。もちろん、各処理モジュールの保持する値のリストが順序付きでない場合でも同様の結果を得ることができる。その場合には、例えば、各処理モジュールは、他の処理モジュールから受信した値のリスト中の値と、自処理モジュールが保持している値のリスト中の値を、すべての組合せについて順番に比較して、各値よりも前に、即ち、上位に、順序付けされる値の個数をカウントして、各値のランクを更新すればよい。
【０１５３】
本実施例４のランク付与方法では、各処理モジュールは、他の処理モジュールから受信した値のリストを保存する必要はなく、自処理モジュールが保持している値のリストに順序を付けるだけで、すべての処理モジュールに共通のランクを付与することができる。また、このランク付与方法は、他の処理モジュールからの値のリストの受信順に影響されないので、処理モジュール間の物理的な接続形態に全く依存しない。したがって、伝送路とランク付与更新回路を多重化することによって、更なる高速化を実現することができる。
【実施例５】
【０１５４】

実施例４では、ランク付与処理は、グローバル項目値番号を付与する順序番号付与処理と類似した手順で行われている。しかし、処理モジュール間の通信を並列に行う場合には、より好適には、順序付け処理は、処理モジュール間の通信を並列に行って個数をカウントする処理と、個数を累計数化する処理の２段階で実行される。例えば、図３６に示された表形式データの記憶構造の一例に基づく合計点のランク付け処理は、図３９に示された本発明の実施例５によるランク付与処理によってランク付けされる。図４０は、本発明の実施例５による順序付け処理のフローチャートである。

【０１５５】

ステップ４００１：処理モジュールＰＭＭ－０は、自処理モジュールで算出された合計点の値４４０、４００及び３７０を値のリスト（４４０，４００，３７０）として保持する。また、個数をカウントするための作業領域rank0、rank1、rank2及びrank3を初期化する。rank0は、自処理モジュールの値のリスト（４４０，４００，３７０）のランクの初期値（０，１，２）を保持する。rank1、rank2及びrank3は、（０，０，０）のように初期化する。図３９の（ａ）は、処理モジュールＰＭＭ－０の初期状態を示している。

【０１５６】

ステップ４００２：処理モジュールＰＭＭ－０は、自処理モジュールの値のリスト（４４０，４００，３７０）を他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３へ送信する。

【０１５７】

ステップ４００３：処理モジュールＰＭＭ－０は、他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３からそれぞれの処理モジュールの値リストを受信する。本例では、図３９の（ｂ）、（ｃ）及び（ｄ）に示されているように、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３からそれぞれの値のリスト（４１０，４４０）、（３８０，４１０，４２０）及び（４４０，４５０）を受信する。

【０１５８】

ステップ４００４：処理モジュールＰＭＭ－０は、他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３から受信した値のリストと自処理モジュールが保持している値のリストを比較して、個数をカウントするための作業領域rank1、rank2及びrank3を更新する。例えば、図３９の（ｂ）は、処理モジュールＰＭＭ－０が処理モジュールＰＭＭ－１から値のリスト（４１０，４４０）を受信したときの処理を示している。

【０１５９】

合計点の値のリストは降順であるため、処理モジュールＰＭＭ－１からの値４１０は、処理モジュールＰＭＭ－０が保持している値４４０よりも下位であり、処理モジュールＰＭＭ－０が保持している値４００よりも上位である。この場合、値４１０は、値４００の直前に挿入されるので、処理モジュールＰＭＭ－０は、値４００に対応したカウント（即ち、rank1の上から２番目）を１だけインクリメントする。また、処理モジュールＰＭＭ－１からの値４４０は、処理モジュールＰＭＭ－０が保持する値４４０と一致し、値４００の直前に挿入されるので、処理モジュールＰＭＭ－０は、値４００に対応したカウントを更に１だけインクリメントする。この結果として、処理モジュールＰＭＭ－０の処理モジュールＰＭＭ－１から受信した値のリストに対するカウントの結果は、図３９の（ｂ）のrank１に示されるように、（０，２，０）となる。

【０１６０】

処理モジュールＰＭＭ－０は、図３９の（ｃ）に示されるように、処理モジュールＰＭＭ－２から受信した値のリスト（３８０，４１０，４２０）と自処理モジュールの値のリスト（４４０，４００，３７０）を比較し、処理モジュールＰＭＭ－２からの値が挿入される直後の自処理モジュールのカウントrank2を１ずつインクリメントする。例えば、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－２からの値３８０に対し、自処理モジュールの値３７０のカウントを１だけインクリメントし、処理モジュールＰＭＭ－２からの値４１０に対し、自処理モジュールの値４００のカウントを１だけインクリメントし、処理モジュールＰＭＭ－２からの値４２０に対し、自処理モジュールの値４００を１だ

けインクリメントする。その結果として、rank2は、（０，２，１）のようになる。

【０１６１】

更に、処理モジュールＰＭＭ－０は、図３９の（ｄ）に示されるように、処理モジュールＰＭＭ－３から受信した値のリスト（４４０，４５０）と自処理モジュールの値のリスト（４４０，４００，３７０）を比較し、処理モジュールＰＭＭ－３からの値が挿入される直後の自処理モジュールのカウントrank3を１ずつインクリメントする。例えば、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－３からの値４４０に対し、自処理モジュールの値４００のカウントを１だけインクリメントし、処理モジュールＰＭＭ－３からの値４５０に対し、自処理モジュールの値４４０のカウントを１だけインクリメントする。その結果として、rank3は、（１，１，０）のようになる。

【０１６２】

ステップ４００５：処理モジュールＰＰＭ－０は、次に、処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３から受信したすべての値のリストによって自処理モジュールＰＭＭ－０に保持されている値のリストのランクの変化を算出するため、rank1、rank2及びrank3を加算する。本例では、図３９の（ｅ）に示されるように、rank1＝（０，２，０）と、rank2＝（０，２，１）と、rank3＝（１，１，０）を加算することにより、（１，５，１）を得る。

【０１６３】

ステップ４００６：処理モジュールＰＰＭ－０は、次に、加算結果を累計数化する。本例では、加算結果＝（１，５，１）を累計数化することにより、累計数＝（１，６，７）が得られる。加算結果＝（１，５，１）は、処理モジュールＰＭＭ－０の値のリスト中の１番目の値４４０の前に１個の値が存在し、１番目の値４４０と２番目の値４００の間に５個の値が存在し、２番目の値４００と３番目の値３７０の間に１個の値が存在することを表している。したがって、値４４０は、処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３からの値のリストの挿入により、１ランク下がり、値４００は、６（＝１＋５）ランク下がり、値３７０は、７（＝６＋１）ランク下がっていることが分かる。

【０１６４】

ステップ４００７：処理モジュールＰＰＭ－０は、最後に、値４４０、４００及び３７０のランクの初期値rank0＝（０，１，２）にステップ４００６で得られたランクの低下数＝（１，６，７）を加算することにより、値のリスト＝（４４０，４００，３７０）に対する最終的なランクrank＝（１，７，９）を算出する。

【０１６５】

上記のステップ４００１～４００７は、他の処理モジュールＰＰＭ－１、ＰＰＭ－２及びＰＰＭ－３においても並列に実行可能であり、処理モジュールＰＭＭ－１の値のリスト＝（４１０，４４０）に対してランクrank＝（５，１）が得られ、処理モジュールＰＭＭ－２の値のリスト＝（３８０，４１０，４２０）に対してランクrank＝（８，５、４）が得られ、処理モジュールＰＭＭ－３の値のリスト＝（４４０，４５０）に対してランクrank＝（１，０）が得られる。本例では、同じ値、例えば、値４４０には同じランク１が付与され、値４４０は３個存在するので、値４４０の次に大きい値４２０には、ランク４（＝１＋３）が付与される。

【０１６６】

以上の処理によって、ランク付与処理が終了する。

【０１６７】

尚、実施例５の説明では、他の処理モジュールから受信する値のリストは、昇順又は降順の順序付けがされていないが、例えば、本例では、他の処理モジュールから受信する値のリストが降順にされているならば、値のリストの比較をより効率的に行うことができる。

【０１６８】

また、合計点の値がＰＭＭ間コンパイルを用いて既にグローバル項目値番号化されている場合には、合計点の値そのものを処理モジュール間で送受信するのではなく、各値に対

応したグローバル項目値番号を処理モジュール間で送受信してもよい。この場合には、値のリストの比較は、グローバル項目値番号を比較することによって実現される。

【実施例６】

【０１６９】

実施例５では、ある処理モジュールに値のリストが存在する場合に、その処理モジュールに対して別の処理モジュールから別の値のリストが送信され、その処理モジュールにおいて値のリスト中の値にランクを付与する処理（順序付け処理）を説明した。例えば、実施例５の例において、処理モジュールＰＭＭ－２に保持されている合計点の値が３８０と４２０と４２０である場合、上記の例では、値のリストとして、（３８０，４２０，４２０）が送信されるが、同じ値が多数回に亘って送信されるような場合には、値と個数のペアのリストを送信した方が効率的である。即ち、（値１，値２，・・・）のような値のリストを送信する代わりに、［（値１，値１の個数），（値２，値２の個数），・・・］のような値と値の個数のペアのリストを送信する。

【０１７０】

そこで、実施例６では、処理モジュール間で値と値の個数のペアのリストを送信して、値にランクを付与する。図４１は、図３９で示した例において、処理モジュール間では値と値の個数のペアのリストが送信され、かつ、処理モジュールＰＭＭ－２から、［（３８０，１），（４２０，２）］が送信されるように変更した例を示す図である。図４２は、本発明の実施例６による順序付け処理のフローチャートである。

【０１７１】

ステップ４２０１：処理モジュールＰＭＭ－０は、自処理モジュールで算出された合計点の値４４０、４００及び３７０を値のリスト（４４０，４００，３７０）として保持する。また、個数をカウントするための作業領域rank0、rank1、rank2及びrank3を初期化する。rank0は、自処理モジュールの値のリスト（４４０，４００，３７０）のランクの初期値（０，１，２）を保持する。rank1、rank2及びrank3は、（０，０，０）のように初期化する。図４１の（ａ）は、処理モジュールＰＭＭ－０の初期状態を示している。

【０１７２】

ステップ４２０２：処理モジュールＰＭＭ－０は、自処理モジュールの値と個数のリスト［（４４０，１），（４００，１），（３７０，１）］を他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３へ送信する。

【０１７３】

ステップ４２０３：処理モジュールＰＭＭ－０は、他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３からそれぞれの処理モジュールの値リストを受信する。本例では、図４１の（ｂ）、（ｃ）及び（ｄ）に示されているように、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３からそれぞれの値と個数のリスト［（４１０、１），（４４０，１）］、［（３８０，１），（４２０，２）］及び［（４４０，１），（４５０，１）］を受信する。

【０１７４】

ステップ４２０４：処理モジュールＰＭＭ－０は、他の処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３から受信した値と個数のリストと自処理モジュールが保持している値のリストを比較して、個数をカウントするための作業領域rank1、rank2及びrank3を更新する。例えば、図４１の（ｂ）は、処理モジュールＰＭＭ－０が処理モジュールＰＭＭ－１から値と個数のリスト［（４１０，１），（４４０，１）］を受信したときの処理を示している。

【０１７５】

合計点の値のリストは降順であるため、処理モジュールＰＭＭ－１からの値４１０は、処理モジュールＰＭＭ－０が保持している値４４０よりも下位であり、処理モジュールＰＭＭ－０が保持している値４００よりも上位である。この場合、値４１０は、値４００の直前に挿入されるので、処理モジュールＰＭＭ－０は、値４００に対応したカウント（即ち、rank1の上から２番目）を値４１０の個数分（本例では１）だけ増やす。また、処理

モジュールＰＭＭ－１からの値４４０は、処理モジュールＰＭＭ－０が保持する値４４０と一致し、値４００の直前に挿入されるので、処理モジュールＰＭＭ－０は、値４００に対応したカウントを更に個数分（本例では１）だけ増やす。この結果として、処理モジュールＰＭＭ－０の処理モジュールＰＭＭ－１から受信した値のリストに対するカウントの結果は、図４１の（ｂ）のrank１に示されるように、（０，２，０）となる。

【０１７６】

処理モジュールＰＭＭ－０は、図４１の（ｃ）に示されるように、処理モジュールＰＭＭ－２から受信した値と個数のリスト［（３８０，１），（４２０，２）］と自処理モジュールの値のリスト（４４０，４００，３７０）を比較し、処理モジュールＰＭＭ－２からの値が挿入される直後の自処理モジュールのカウントrank2を挿入される値の個数分ずつ増やす。例えば、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－２からの値３８０に対し、自処理モジュールの値３７０のカウントを１だけ増やし、処理モジュールＰＭＭ－２からの値４２０に対し、自処理モジュールの値４００のカウントを２だけ増やす。その結果として、rank2は、（０，２，１）のようになる。

【０１７７】

更に、処理モジュールＰＭＭ－０は、図４１の（ｄ）に示されるように、処理モジュールＰＭＭ－３から受信した値のリスト［（４４０，１），（４５０，１）］と自処理モジュールの値のリスト（４４０，４００，３７０）を比較し、処理モジュールＰＭＭ－３からの値が挿入される直後の自処理モジュールのカウントrank3を挿入される値の個数分ずつ増やす。例えば、処理モジュールＰＭＭ－０は、処理モジュールＰＭＭ－３からの値４４０に対し、自処理モジュールの値４００のカウントを１だけ増やし、処理モジュールＰＭＭ－３からの値４５０に対し、自処理モジュールの値４４０のカウントを１だけ増やす。その結果として、rank3は、（１，１，０）のようになる。

【０１７８】

ステップ４２０５：処理モジュールＰＰＭ－０は、次に、処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３から受信したすべての値のリストによって自処理モジュールＰＭＭ－０に保持されている値のリストのランクの変化を算出するため、rank1、rank2及びrank3を加算する。本例では、図４１の（ｅ）に示されるように、rank1＝（０，２，０）と、rank2＝（０，２，１）と、rank3＝（１，１，０）を加算することにより、（１，５，１）を得る。

【０１７９】

ステップ４２０６：処理モジュールＰＰＭ－０は、次に、加算結果を累計数化する。本例では、加算結果＝（１，５，１）を累計数化することにより、累計数＝（１，６，７）が得られる。加算結果＝（１，５，１）は、処理モジュールＰＭＭ－０の値のリスト中の１番目の値４４０の前に１個の値が存在し、１番目の値４４０と２番目の値４００の間に５個の値が存在し、２番目の値４００と３番目の値３７０の間に１個の値が存在することを表している。したがって、値４４０は、処理モジュールＰＭＭ－１、ＰＭＭ－２及びＰＭＭ－３からの値のリストの挿入により、１ランク下がり、値４００は、６（＝１＋５）ランク下がり、値３７０は、７（＝６＋１）ランク下がっていることが分かる。

【０１８０】

ステップ４２０７：処理モジュールＰＰＭ－０は、最後に、値４４０、４００及び３７０のランクの初期値rank0＝（０，１，２）にステップ４００６で得られたランクの低下数＝（１，６，７）を加算することにより、値のリスト＝（４４０，４００，３７０）に対する最終的なランクrank＝（１，７，９）を算出する。

【０１８１】

上記のステップ４２０１～４２０７は、他の処理モジュールＰＰＭ－１、ＰＰＭ－２及びＰＰＭ－３においても並列に実行可能であり、処理モジュールＰＭＭ－１の値のリスト＝（４１０，４４０）に対してランクrank＝（５，１）が得られ、処理モジュールＰＭＭ－２の値のリスト＝（３８０，４１０，４２０）に対してランクrank＝（８，５、４）が得られ、処理モジュールＰＭＭ－３の値のリスト＝（４４０，４５０）に対してランクra

nk＝（１，０）が得られる。本例では、同じ値、例えば、値４４０には同じランク１が付与され、値４４０は３個存在するので、値４４０の次に大きい値４２０には、ランク４（＝１＋３）が付与される。以上の処理によって、ランク付与処理が終了する。

【０１８２】

尚、実施例６の説明では、他の処理モジュールから受信する値と個数のリストは、昇順又は降順の順序付けがされていないが、例えば、本例では、他の処理モジュールから受信する値と個数のリストが値に関して降順にされているならば、値のリストの比較をより効率的に行うことができる。

【０１８３】

また、合計点の値がＰＭＭ間コンパイルを用いて既にグローバル項目値番号化されている場合には、合計点の値そのものを処理モジュール間で送受信するのではなく、各値に対応したグローバル項目値番号と個数のリストを処理モジュール間で送受信してもよい。例えば、上記の例では、処理モジュールＰＭＭ－０が他の処理モジュールへ送信する値４４０、４００及び３７０は、グローバル項目値番号１、４及び６に対応しているので、処理モジュールＰＰＭ－０は、［（４４０，１），（４００，１），（３７０，１）］の代わりに、［（１，１），（４，１），（６，１）］を送信する。この場合には、値の比較は、グローバル項目値番号を比較することによって実現される。

【０１８４】

更に、値がグローバル項目値番号化されている場合には、各処理モジュールは、自処理モジュールに保持されている値と値の個数のリスト、或いは、グローバル項目値番号と個数のリストを送信するのではなく、グローバル項目値番号の順に値の個数を並べた「個数のリスト」を送信してもよい。上記の例では、処理モジュールＰＭＭ－０は、（０，１，０，０，１，０，１）を送信する。この場合、非零の値がリストの何番目に出現しているかを検出することによって、対応したグローバル項目値番号を取得することができるので、グローバル項目値番号の比較を簡単に行うことができる。

【０１８５】

最後に、上記の実施例１～６の説明では詳細に記述していない事項について解説する。

【０１８６】

［ローカル・ソート処理］

ローカル・ソート処理は、実施例１の図９を参照して説明したように、グローバルな集計処理の一部として、或いは、グローバルなソート処理の一部として実行される処理である。本実施の形態では、ローカル・ソート処理が各処理モジュールで独立に行われるので、このローカル・ソート処理を高速化することにより、集計処理の処理速度を高速化することができる。

【０１８７】

以下、このローカル・ソート処理について説明する。尚、ローカル・ソート処理は、図４３に示されるように、コンパイル処理が終了した状態から処理が開始されるものとして説明する。図４４は、ローカル・ソート処理のフローチャートである。図４４に示すように、各ＰＭＭは、ソートすべき項目に関する値リストVLと同一のサイズの、存在数配列の領域を生成し（ステップ４４０１）、領域中の各値に初期値「０」を与える（ステップ４４０２）。図４５は、「年齢」という項目について、それぞれのＰＭＭにおいて、値リストVLと同一のサイズを有する領域が作られ、それぞれに初期値「０」が与えられた状態を示す。

【０１８８】

次いで、各ＰＭＭは、存在数配列のそれぞれに対するカウントアップ処理を実行する（ステップ４４０３）。より詳細には、各ＰＭＭは、順序集合配列OrdSetの値を参照して、ソートすべき項目のポインタ配列VNoの値を特定する（ステップ４４１１）。次いで、各ＰＭＭは、存在数配列中、当該ポインタ配列VNoの値に示される位置の値をカウントアップする（ステップ４４１２）。このような処理が、順序集合配列OrdSetの末尾まで繰り返される（ステップ４４１３、４４１４参照）。

【０１８９】
図４６は、各ＰＭＭにおけるカウントアップの一例を示す図である。たとえば、ＰＭＭ－０において、順序集合配列OrdSetの要素「０」が示す位置の、年齢のポインタ配列VNoの値は「０」である。したがって、存在数配列の「第０番目」の位置、つまり、先頭の位置にある値を、「０」から「１」にカウントアップする。他のＰＭＭにおいても、同様の処理が実行されていることが理解できるであろう。
【０１９０】
カウントアップ処理が終了すると、図４７に示すように、各ＰＭＭは、存在数配列の要素を累計して、当該存在数配列を累計数配列に変換する（ステップ４７０１）。累計数配列の要素である累計数は、項目値を指し示すレコードの数を示す存在数を考慮して、当該累計数が配置されている位置の項目値を指し示すレコードの先頭の位置を示すようになっている。具体的には、各ＰＭＭが、配列の位置を示すパラメータ「ｉ」を初期化して（ステップ４７１１）、パラメータが示す存在数配列中の値を取り出し（ステップ４７１２）、パラメータ「ｉ」が示す位置より、後ろの位置、つまり、「ｉ＋１」、「ｉ＋２」、・・・の位置の存在数配列の値に、ステップ４７１２で取り出された値を、それぞれ加算する（ステップ４７１３）。ステップ４７１２、４７１３に示す処理を、値リストVLの要素（項目値）の個数だけ繰り返せばよい（ステップ４７１４、４７１５参照）。
【０１９１】
このようにして、たとえば、図４８に示すような累計数配列を得ることができる。さらに、各ＰＭＭは、後でＰＭＭ全体における順位を格納するための配列GVNo、GOrd’及びOrdSet’のための領域も作られる（ステップ４７０２）。これら配列のサイズは、それぞれ、値リストVLのサイズと一致する。
【０１９２】
次に、各ＰＭＭにおけるローカルなソート処理が実行される。図４９に示すように、各ＰＭＭは、順序集合配列OrdSetの値を取り出し（ステップ４９０１）、次いで、ポインタ配列VNo中、配列OrdSetの値が指し示す位置の値（ポインタ値）を特定する（ステップ４９０２）。その後、各ＰＭＭは、ソートすべき項目のグローバル項目値番号配列GVNo中、ポインタ配列VNoの値が示す位置の値を取得する（ステップ４９０３）。この値は、後述する値の格納処理に利用される。その一方、上記累計数配列においても、ポインタ配列VNoが示す位置の値が取得される（ステップ４９０４）。この値は、後述する値の格納処理において、配列中の位置を指定するために利用される。
【０１９３】
次に値の格納処理が実行される。各ＰＭＭは、先に生成した配列GVNo中、ステップ４９０４で取得された累計数配列の値が示す位置に、ステップ４９０２で取得された、ソートすべき項目に関するGVNoの値を配置する（ステップ４９０５）。また、各ＰＭＭは、配列GOrd’、OrdSet’中、ステップ４９０４で取得された累計数配列の値が示す位置に、グローバル順序集合配列GOrd及び順序集合配列OrdSetの値を、それぞれ配置する（ステップ４９０６）。次いで、処理に用いられた累計数配列の値がインクリメントされる（ステップ４９０７）。
【０１９４】
上記ステップ４９０１～４９０７の処理が、配列OrdSet中の全ての値について、順次実行される（ステップ４９０８、４９０９参照）。
【０１９５】
図５０及び図５１は、各ＰＭＭにおいてローカルなソート処理が実行されている状態の例を示す図である。たとえば、ＰＭＭ－０に関して、図５０においては、配列OrdSetの値「０」の取り出し（ステップ４９０１参照）、当該OrdSetの値「０」が示す位置の、配列VNoの値「０」の特定（ステップ４９０２参照）、当該配列VNoの値「０」が示す位置の、配列GVNoの値「１」の取得（ステップ４９０３）、および、配列VNoの値「０」が示す位置の、累計数配列の値「０」の取得（ステップ４９０４）が実行されていることが理解できるであろう。また、累計数配列の取得の後、当該累計数配列の値が、「０」から「１」

になっていることもわかる（ステップ４９０７参照）。

【０１９６】

また、ＰＭＭ－０に関して、図５１において、ステップ４１０３で取得された累計数配列の値の示す位置における、配列GVNo、GOrd’及びOrdSet’への、項目「年齢」に関する配列GVNoの値「１」、並びに、配列GOrdの値「０」及び配列OrdSetの値「０」の配置（ステップ４９０５、４９０６）が示されていることが理解できるであろう。他のＰＭＭについても、図５０及び５１において、同様にステップ４９０１～４９０５に示す処理が実行されていることがわかる。

【０１９７】

以上のローカルな（つまり各ＰＭＭにおける）ソート処理により、図５２に示すような配列を得ることができる。尚、図５０乃至５２において、図中の「昇順２」とは、グローバル・レコード番号GOrd’は、グローバル項目値番号GVNo’が同一値である範囲内で「昇順」になるということを意味する。

【０１９８】

ここで説明したローカル・ソート処理は、比較演算を行わないという優れた性質がある。一般的に、比較を行うソートは、データ数をｎとするとき、0(n*log(n))の処理量が発生し、比較を行わないソートでは、処理量は0(n)である。比較を行わないカウンティングソートは、数え上げ、累計数化、及び、転送のおよそ３つの段階を含む。処理ステップは、全てのデータが異なる場合、３ｎになる。そこで、ｎ個の重複しないデータがあって、m台のコンピュータがある場合、ｎ個のデータをm分割して、各分割部分をローカルにソートし、それをグローバル・ソートで統合するモデルでは、グローバル・ソートのおおよそのステップは

（ｍ－１）＊（２＊ｎ／ｍ）

になる。最初の項（ｍ－１）は、各コンピュータが他のコンピュータから受け取って処理しなければならない回数を表し、２番目の項（２＊ｎ／ｍ）は、ｎ／ｍ個の２つの昇順リストを比較する際に、平均的に発生する比較の回数である。mが大きければ、

２＊ｎ

となり、グローバル・ソートのステップ数は、0(n)になる。即ち、比較を行うソート0(n*log(n))よりも効率がよい。これは、昇順のリストを比較することにより、効率化されたためと考えることができる。一方、mが消えていることは、コンピュータの台数が増えても、グローバル・ソートにおける１台あたりの処理量は変わらないことを意味している。

【０１９９】

［ローカル・ソート処理の他の実施形態］

上述のローカル・ソート処理は、各処理モジュールを並列に動作させ得る点で優れている。しかし、ローカル・ソート処理は、他の方式でも実現できる。例えば、コンピュータの台数mがデータの個数ｎに匹敵する場合には、上述の順序番号付与処理の考え方を用いてローカル・ソート処理を実現してもよい。

【０２００】

例えば、図９を参照して説明した「年齢」と「性別」でソートする例に関して、この他の実施形態によるローカル・ソート処理を説明する。年齢と性別でソートする例では、各ＰＭＭにおいて、レコード毎に性別GVNo、年齢GVNo、及び、GOrdの３次元の配列を作り、この３次元の配列に一気に順序番号を付与すれば、上述のローカル・ソート処理と同じ結果が得られる。図５３は、順序番号付与処理を使用するローカル・ソート処理の説明図である。

【０２０１】

最初に、図５３（Ａ）に示されるように、OrdSetの各要素について、性別のGVNo、年齢のGVNo及びGOrdからなる３次元配列を作成する。以下の説明では、OrdSet＝ｉの３次元配列を、Ａ［ｉ］＝（ａ，ｂ，ｃ）のように表現することにする。本例では、

Ａ［０］＝（１，２，０）

Ａ［１］＝（０，０，１）

Ａ［２］＝（１，１，２）
である。
【０２０２】
次に、図５３（Ｂ）に示されるように順序番号を初期化する。
次に、図５３（Ｃ）に示されるように、順序番号を付与する。本例では、Ａ［０］をOrdSet１へ送り、Ａ［１］をOrdSet２へ送り、Ａ［２］をOrdSet０へ送り、自分の保持する３次元配列と送られてきた配列を比較し、順序番号を付与する。
【０２０３】
更に、図５３（Ｄ）に示されるように、Ａ［０］をOrdSet２へ送り、Ａ［１］をOrdSet０へ送り、Ａ［２］をOrdSet１へ送り、自分の保持する３次元配列と送られてきた配列を比較し、順序番号を付与する。
【０２０４】
このような順序番号付与処理の結果として、図５３（Ｅ）に示されるような結果が得られる。図５３（Ｆ）には、順序番号の順に入れ替えた結果が示されている。図５３（Ｆ）に示されている結果は、図９に示されているローカル・ソートの結果と一致している。
【０２０５】
［ＳＩＭＤ型並列処理］
並列化のアルゴリズムが稚拙である場合、ＳＩＭＤを採用して所望の結果を得るためのプログラムの開発が困難であり、開発できたとしても、そのプログラムの自由度は低い。そこで、ＳＩＭＤを採用するためには、ＳＩＭＤに適した優れたアルゴリズムを開発する必要がある。この点、本実施の形態によるアルゴリズムは、
（１）処理の実行にあたって条件分岐がない。但し、検索処理の場合には、条件分岐が行われる可能性があるが、単純な条件分岐である。
（２）昇順のリストを相互に比較するなど、１つの命令で実行できる処理（ステップ数、クロック数）の占める割合が高い。
（３）すべての処理モジュールが平等に同じ役割を持つ。処理モジュール毎に違う役割があると、単一命令で処理を実現できない。
という点でデータ構造及びアルゴリズムが優れている。したがって、本実施の形態では、ＳＩＭＤを採用する際にプログラムが簡単化され、プログラムの開発の容易性やプログラムの高い自由度を確保することができる。
【０２０６】
［システム構成］
本発明にかかる情報処理システムは、例えば、フロントエンドとなる端末装置と、リング状のチャネルを介して接続され、端末装置からの命令を、それぞれのＰＭＭが受理することにより、ＰＭＭにおいて、上述したコンパイル、ソート、集計の処理が実行できる。また。各ＰＭＭはパケットを何れかのバスを利用して送出すればよく、ＰＭＭ間の同期等を外部から制御する必要もない。
【０２０７】
また、制御装置には、コンパイル、ソートなどの繰り返し演算のためのハードウェア構成を備えたアクセラレータチップのほか、これに加えて、汎用ＣＰＵを含めても良い。汎用ＣＰＵは、端末装置からチャネルを介して伝達された命令を解釈し、アクセラレータチップに必要な指示を与えることができる。
【０２０８】
さらに、制御装置、特に、その中のアクセラレータチップには、順序集合配列、グローバル順序集合配列など作業に必要な種々の配列を収容するためのレジスタ群が設けられているのが望ましい。これにより、いったん、メモリからレジスタ上に処理に必要な値をロードしてしまえば、コンパイル、ソート、及び、集計にかかる上述した処理演算中には、制御装置はメモリにアクセスすることなく、レジスタから値を読み出し、或いは、レジスタに値を書き込めばよい。これにより、メモリアクセスの回数を著しく減じる（演算処理前のロード、および、処理結果の書き込み）ことができ、処理時間を著しく短縮すること

が可能となる。

【０２０９】

本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。

【０２１０】

前記実施の形態においては、ＰＭＭを、一方が時計回りにパケットを伝送する第１のバス（第１の伝送路）、他方が反時計回りにパケットを伝送する第２のバス（第２の伝送路）にて、リング状に接続している。このような構成により、パケット伝送の遅延時間などを均一化することができるため有利である。しかしながら、これに限定されず、バス型など他の形態の伝送路を採用しても良い。

【０２１１】

また、本実施の形態においては、メモリ、インタフェースおよび制御回路を有するＰＭＭを利用しているが、これに限定されるものではなく、パーソナルコンピュータ、サーバなどを、ローカルな表形式データを分掌する情報処理ユニットとして、ＰＭＭの代わりに利用しても良い。或いは、単一のパーソナルコンピュータやサーバが、複数の情報処理ユニットを保持するような構成を採用しても良い。これらの場合でも、情報処理ユニットが、レコードの順位を示す値を受理し、グローバル順序集合配列GOrdを参照することにより、レコードを特定することができる。また、グローバル値番号配列を参照することにより、項目値を特定することも可能である。

【０２１２】

また、情報処理ユニット間の伝送路も、いわゆるネットワーク型やバス型を採用しても良い。

【０２１３】

単一のパーソナルコンピュータに複数の情報処理ユニットを設けるような構成を採用することで、以下のように、本発明を利用することができる。たとえば、札幌支社、東京支社、福岡支社の３つの表形式データを用意し、通常は、各支社の単位で、検索、集計、ソートなどを実行する。さらに、３つの支社を統合したグローバルな表形式データを考えて、各支社の表形式データが、全体表のうちの部分表であるとみなし、グローバルな表形式データに関する検索、ソートおよび集計を実現することができる。

【０２１４】

無論、複数のパーソナルコンピュータをネットワークにて接続した場合にも、同様に、パーソナルコンピュータにて分掌されるローカルな表形式データに関する処理、および、グローバルな表形式データに関する処理を実現することもできる。

【図面の簡単な説明】

【０２１５】

【図１】従来のデータ管理機構の説明図である。

【図２】従来のデータ管理機構の説明図である。

【図３】本発明の実施の形態にかかる情報処理システムの概略を示すブロックダイヤグラムである。

【図４】本発明の実施の形態にかかるＰＭＭの構造の一例を示す図である。

【図５】表形式データの一例の説明図である。

【図６】従来の表形式データの記憶構造の説明図である。

【図７】本発明の実施の形態にかかる表形式データの記憶構造の一例の説明図である。

【図８】本発明の実施の形態にかかる集計処理のフローチャートである。

【図９】ローカル・ソート処理の結果の説明図である。

【図１０】本発明の実施の形態にかかる順序番号付与処理の説明図である。

【図１１】本発明の実施の形態にかかるローカル次元値番号付与処理の説明図である。

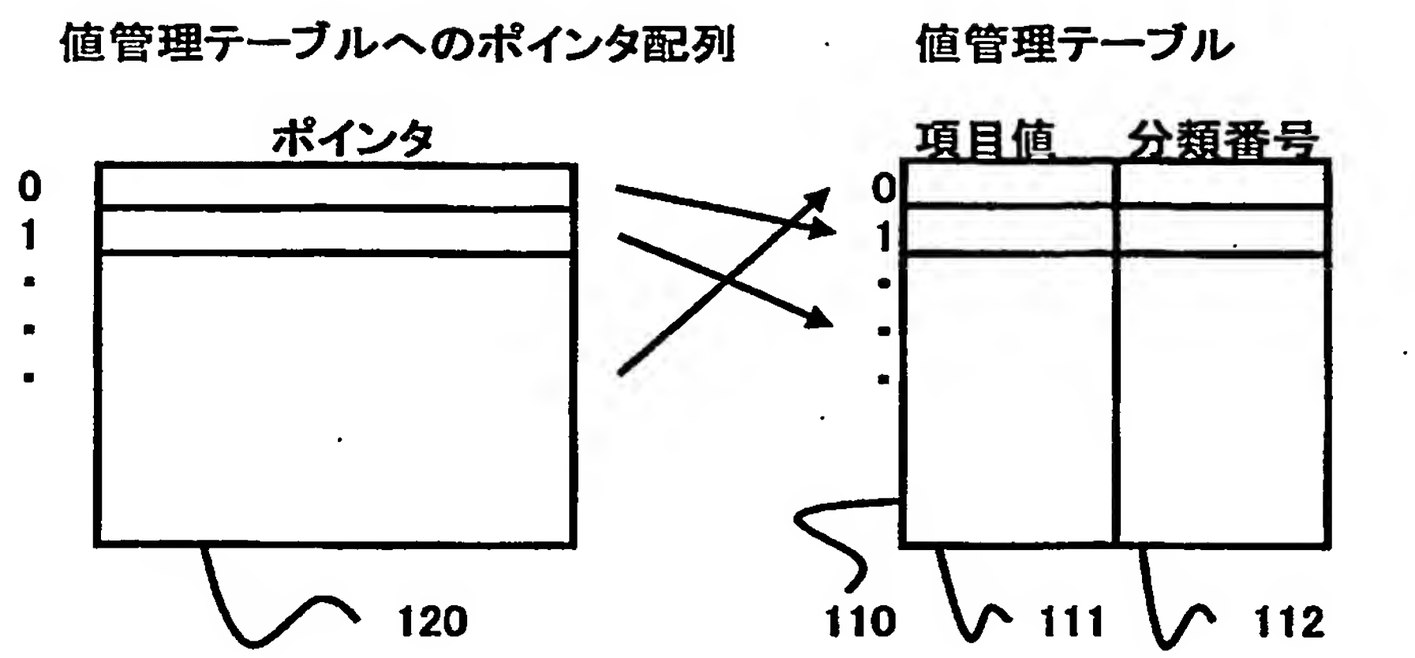
【図１２】本発明の実施にかかるグローバル次元値番号付与処理の説明図である。
【図１３】グローバル次元値番号付与処理の結果の説明図である。
【図１４】本発明の実施例１によるローカル集計処理の説明図である。
【図１５】本発明の実施例１によるローカル集計処理の説明図である。
【図１６】本発明の実施例１による第１のグローバル集計方法の説明図である。
【図１７】本発明の実施例１によるグローバル集計結果算出の説明図である。
【図１８】本発明の実施例１によるグローバル集計値の重複排除処理の説明図である。
【図１９】本発明の実施例１による結果テーブル生成処理の説明図である。
【図２０】本発明の実施例１による結果テーブルの説明図である。
【図２１】本発明の実施例１による第２のグローバル集計方法の説明図である。
【図２２】本発明の実施例１による順序番号付与方法のフローチャートである。
【図２３】本発明の実施例１による順序番号付与方法の説明図である。
【図２４】本発明の実施例１による順序番号付与方法の説明図である。
【図２５】本発明の実施例２によるローカル集計処理の説明図である。
【図２６】本発明の実施例２によるローカル集計処理の説明図である。
【図２７】本発明の実施例２による第１のグローバル集計方法の説明図である。
【図２８】本発明の実施例２によるグローバル集計結果算出の説明図である。
【図２９】本発明の実施例２によるグローバル集計値の重複排除処理の説明図である。
【図３０】本発明の実施例２による結果テーブル生成処理の説明図である。
【図３１】本発明の実施例２による結果テーブルの説明図である。
【図３２】本発明の実施例２による第２のグローバル集計方法の説明図である。
【図３３】本発明の実施例３による表形式データの記憶構造の一例の説明図である。
【図３４】本発明の実施例３によるカウント処理の説明図である。
【図３５】本発明の実施例３による他のカウント処理の説明図である。
【図３６】本発明の実施例４による表形式データの記憶構造の一例の説明図である。
【図３７】本発明の実施例４による順序付け処理の説明図である。
【図３８】本発明の実施例４による順序付け処理のフローチャートである。
【図３９】本発明の実施例５によるランク付与処理の説明図である。
【図４０】本発明の実施例５による順序付け処理のフローチャートである。
【図４１】本発明の実施例６によるランク付与処理の説明図である。
【図４２】本発明の実施例６による順序付け処理のフローチャートである。
【図４３】コンパイル処理終了時の状態の一例の説明図である。
【図４４】本発明の実施の形態にかかるローカル・ソート処理のフローチャートである。
【図４５】ローカル・ソート処理の初期状態の一例の説明図である。
【図４６】各ＰＭＭにおけるカウントアップ処理の一例の説明図である。
【図４７】累計数配列作成処理の一例の説明図である。
【図４８】累計数配列の一例の説明図である。
【図４９】ローカル・ソート処理の詳細なフローチャートである。
【図５０】各ＰＭＭにおけるローカル・ソート処理の実行状態の一例の説明図である。
【図５１】各ＰＭＭにおけるローカル・ソート処理の実行状態の一例の説明図である。
【図５２】各ＰＭＭにおけるローカル・ソート処理の結果の一例の説明図である。
【図５３】（Ａ）から（Ｆ）はローカル・ソート処理の他の実施形態の説明図である。

【符号の説明】
【０２１６】

３２　　ＰＭＭ
３４　　第１のバス
３６　　第２のバス
４０　　制御回路
４２　　バスＩ／Ｆ
４４　　メモリ
４６　　バンク

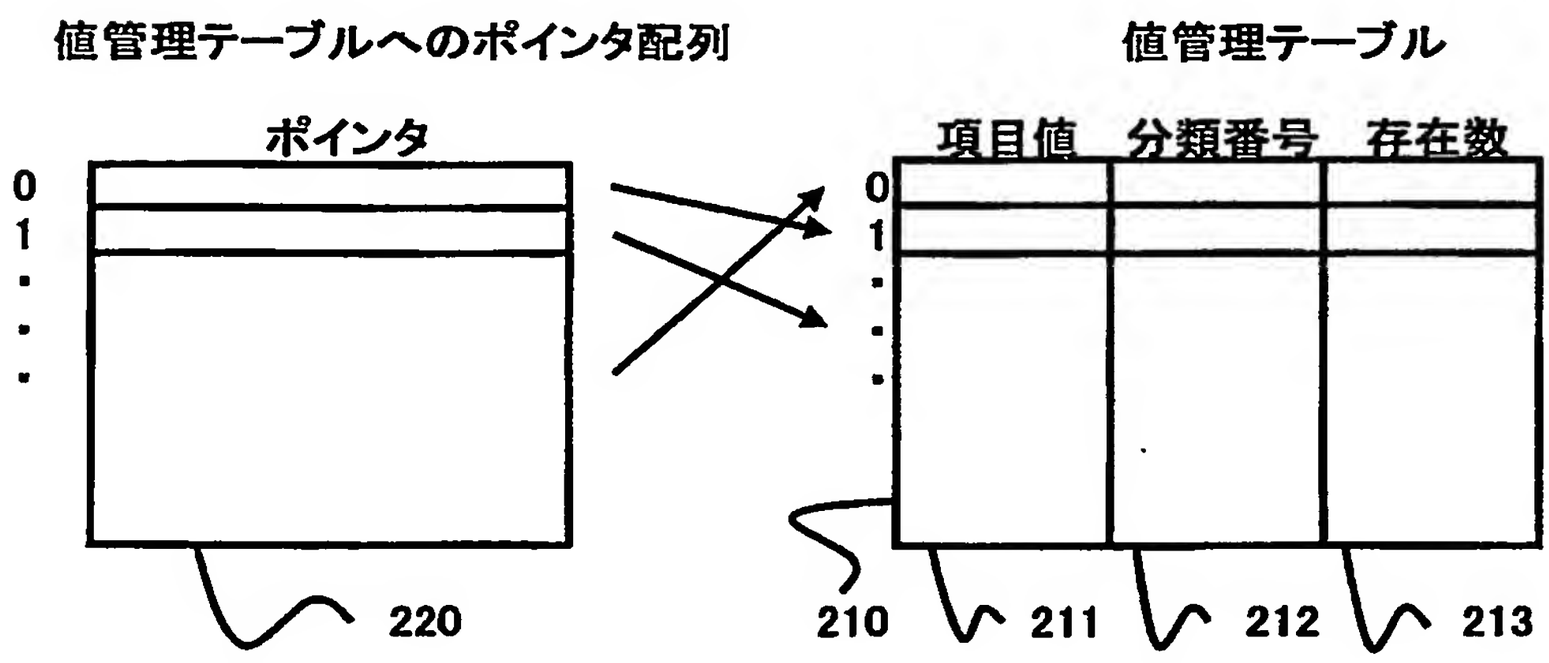
【書類名】図面
【図１】

図1



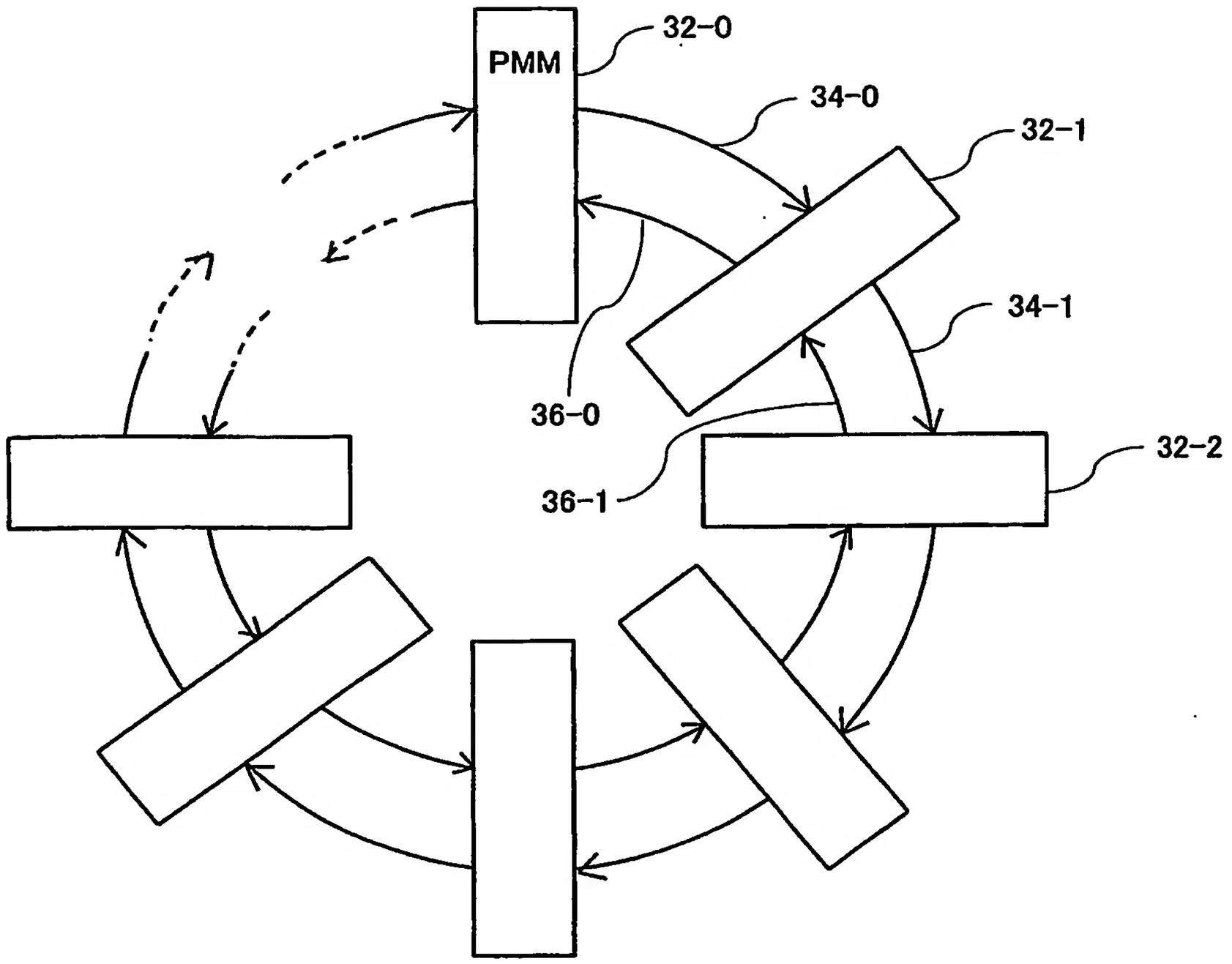
【図２】

図2



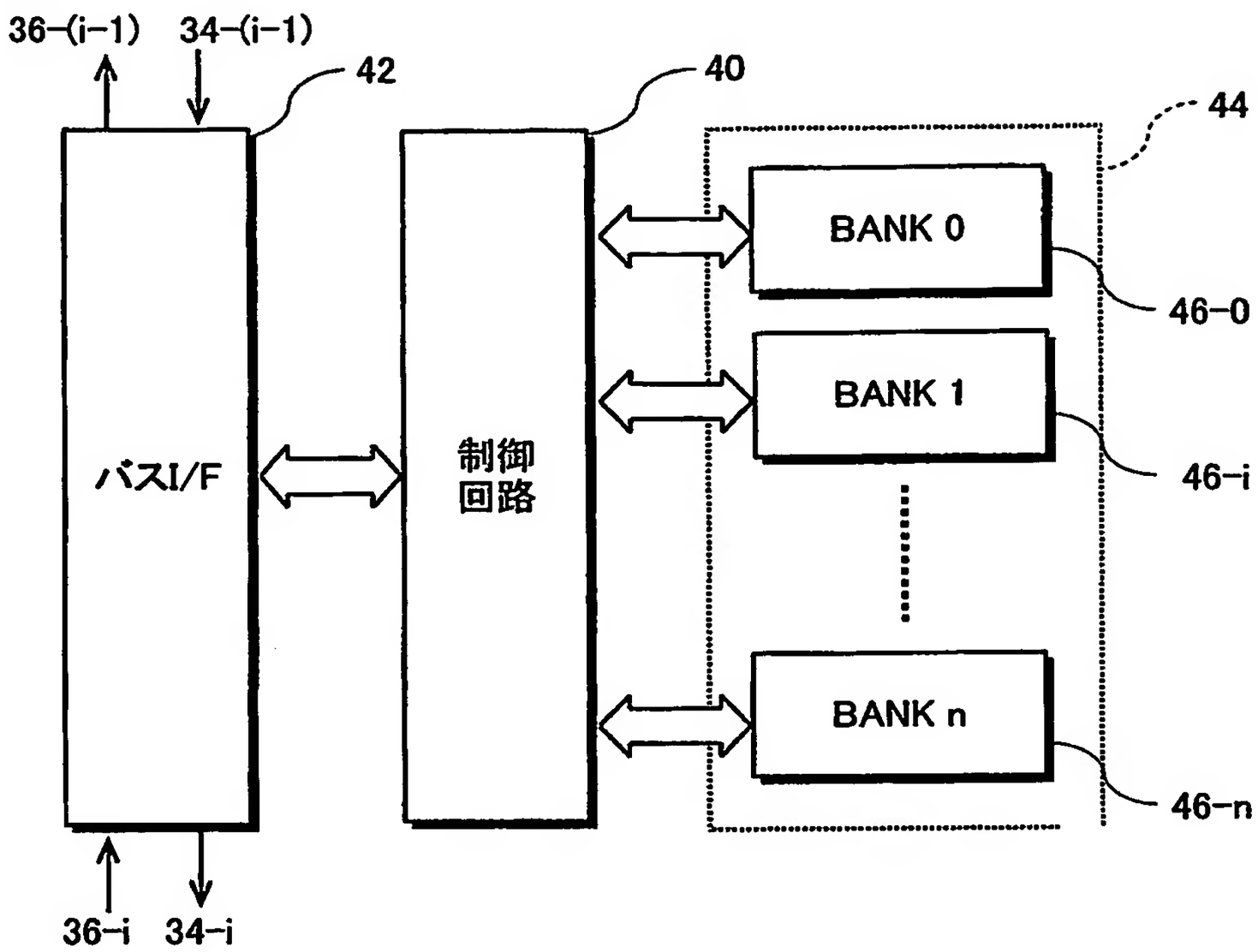
【図３】

図3



【図４】

図4



【図５】

図5

レコード番号	別	年齢	身長(cm)
0	女	3	78
1	男	1	82
2	女	2	69
3	女	1	82
4	男	3	91
5	女	1	76
6	女	1	78
7	女	2	84
8	男	3	87
9	女	3	80

【図６】

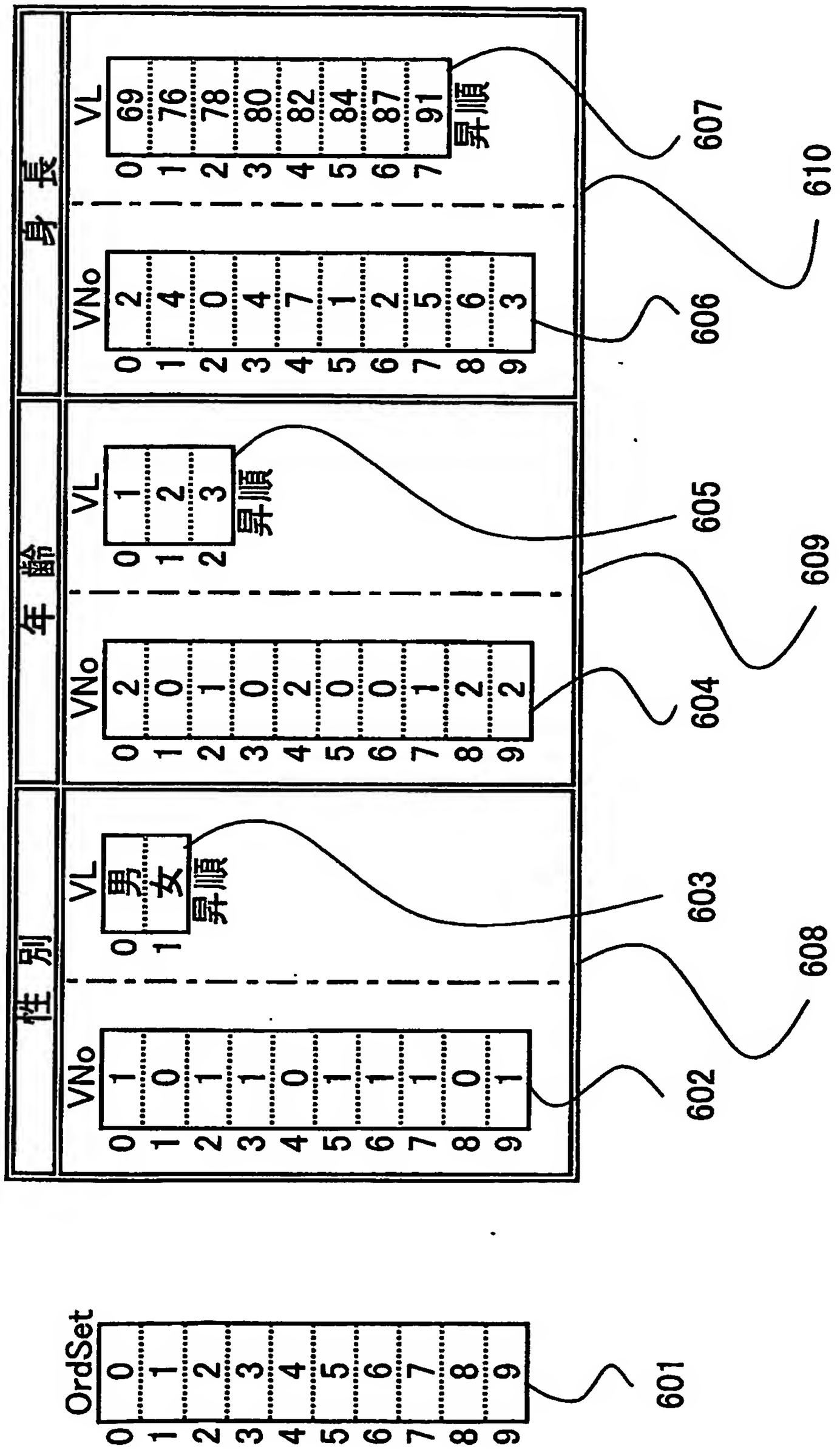

性別
男
女
昇順
VL
VNo
0 1 2 3 4 5 6 7 8 9
年齢
昇順
VL
VNo
0 1 2 3 4 5 6 7 8 9
身長
昇順
VL
VNo
0 1 2 3 4 5 6 7 8 9
OrdSet
0 1 2 3 4 5 6 7 8 9
601
602
603
604
605
606
607
608
609
610

図6

【図７】

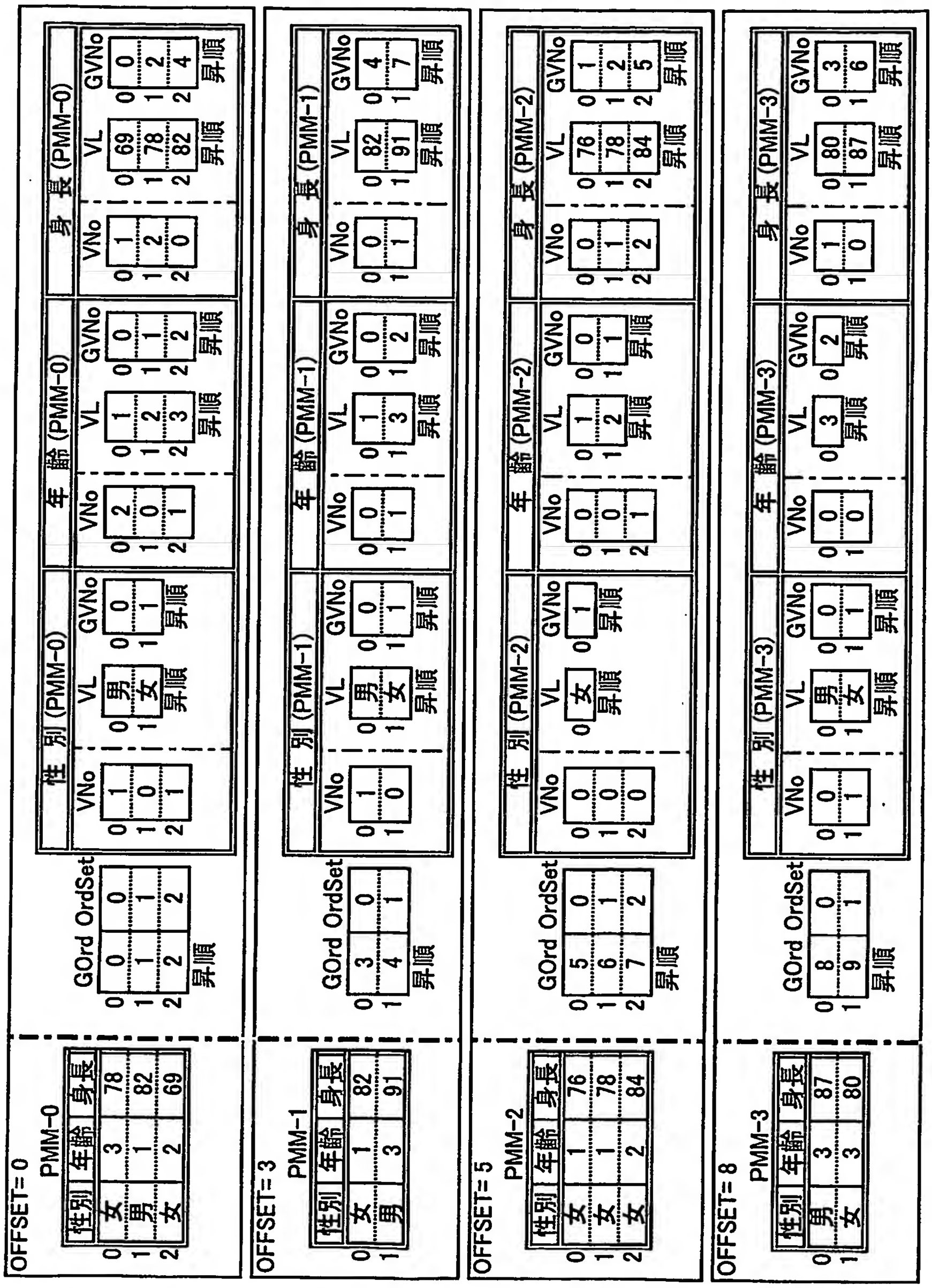
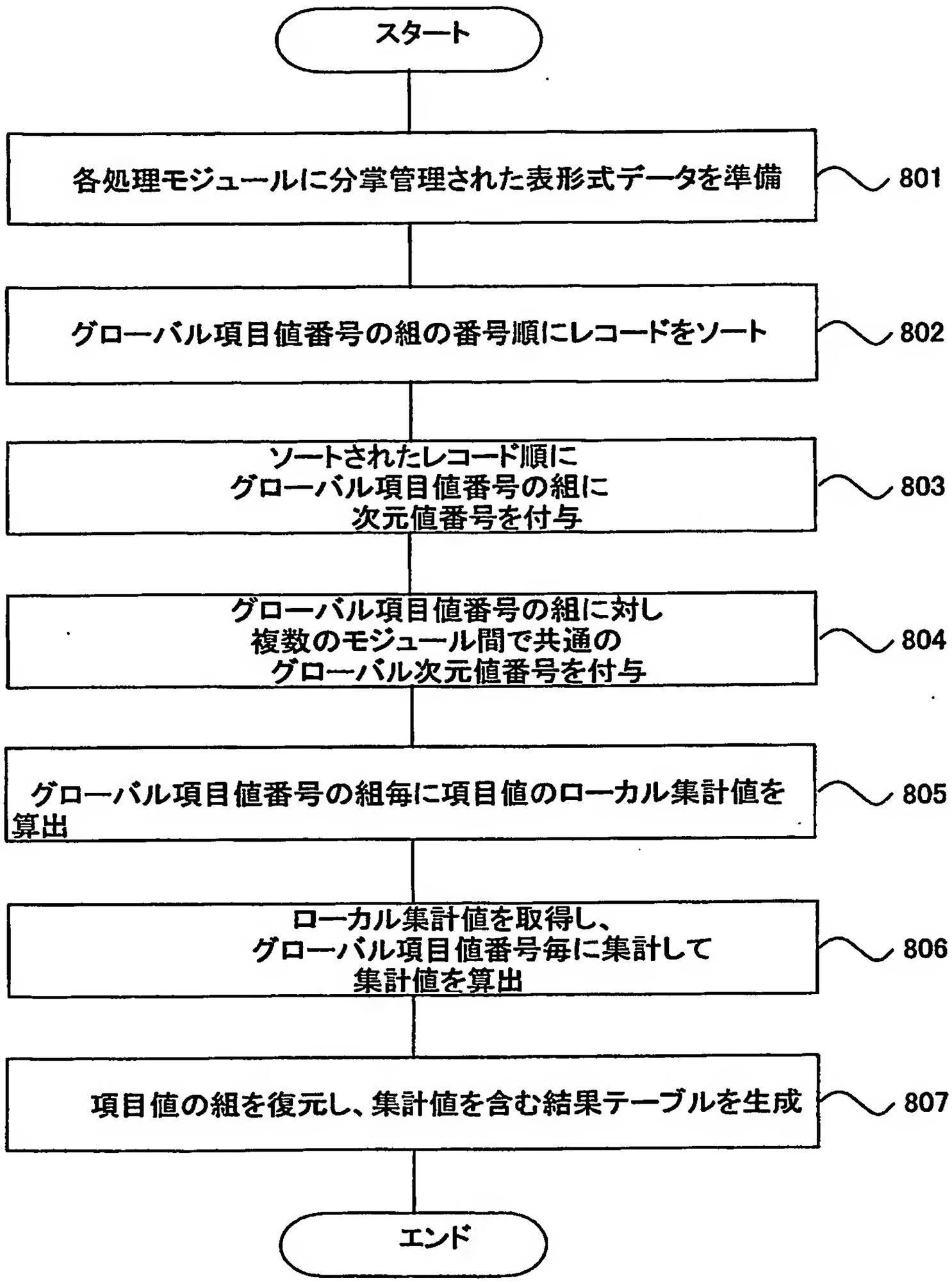

身長 (PMM-0)
VNo
VL
GVNo
昇順
年齢 (PMM-0)
性別 (PMM-0)
GOrd OrdSet
身長 (PMM-1)
年齢 (PMM-1)
性別 (PMM-1)
身長 (PMM-2)
年齢 (PMM-2)
性別 (PMM-2)
身長 (PMM-3)
年齢 (PMM-3)
性別 (PMM-3)
男
女
OFFSET= 0
PMM-0
性別 年齢 身長
女 3 78
男 1 82
女 2 69
OFFSET= 3
PMM-1
性別 年齢 身長
女 1 82
男 3 91
OFFSET= 5
PMM-2
性別 年齢 身長
女 1 76
女 1 78
女 2 84
OFFSET= 8
PMM-3
性別 年齢 身長
男 3 87
女 3 80

図7

【図８】

図8


スタート
各処理モジュールに分掌管理された表形式データを準備
801
グローバル項目値番号の組の番号順にレコードをソート
802
ソートされたレコード順に
グローバル項目値番号の組に
次元値番号を付与
803
グローバル項目値番号の組に対し
複数のモジュール間で共通の
グローバル次元値番号を付与
804
グローバル項目値番号の組毎に項目値のローカル集計値を
算出
805
ローカル集計値を取得し、
グローバル項目値番号毎に集計して
集計値を算出
806
項目値の組を復元し、集計値を含む結果テーブルを生成
807
エンド

【図９】

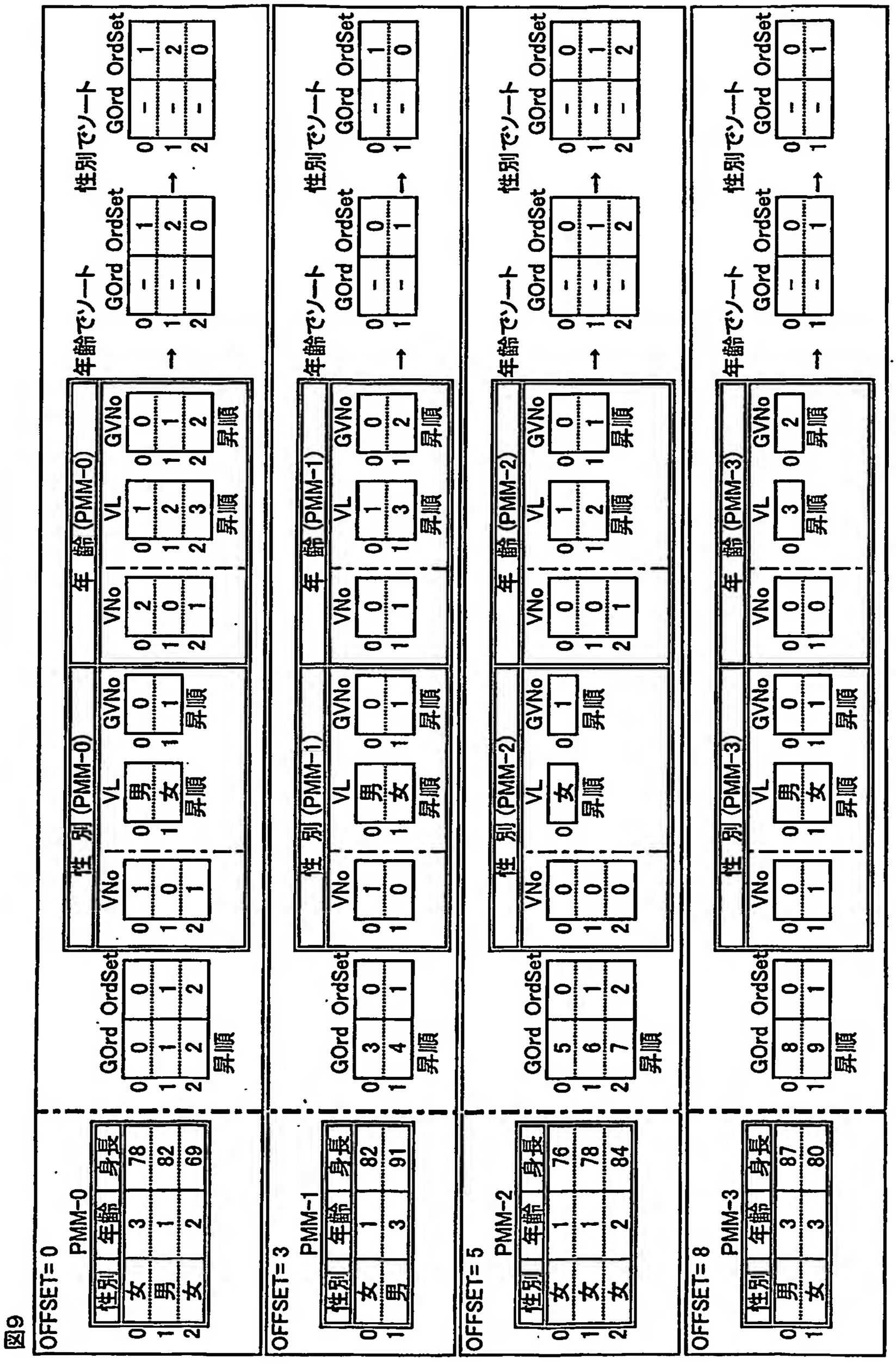


図9
OFFSET= 0
PMM-0
性別
年齢
身長
女 3 78
男 1 82
女 2 69
OFFSET= 3
PMM-1
性別
年齢
身長
女 1 82
男 3 91
OFFSET= 5
PMM-2
性別
年齢
身長
女 1 76
女 1 78
女 2 84
OFFSET= 8
PMM-3
性別
年齢
身長
男 3 87
女 3 80
性別(PMM-0)
年齢(PMM-0)
性別(PMM-1)
年齢(PMM-1)
性別(PMM-2)
年齢(PMM-2)
性別(PMM-3)
年齢(PMM-3)
VNo
VL
GVNo
GOrd
OrdSet
男
女
昇順
年齢でソート
性別でソート

【図１０】

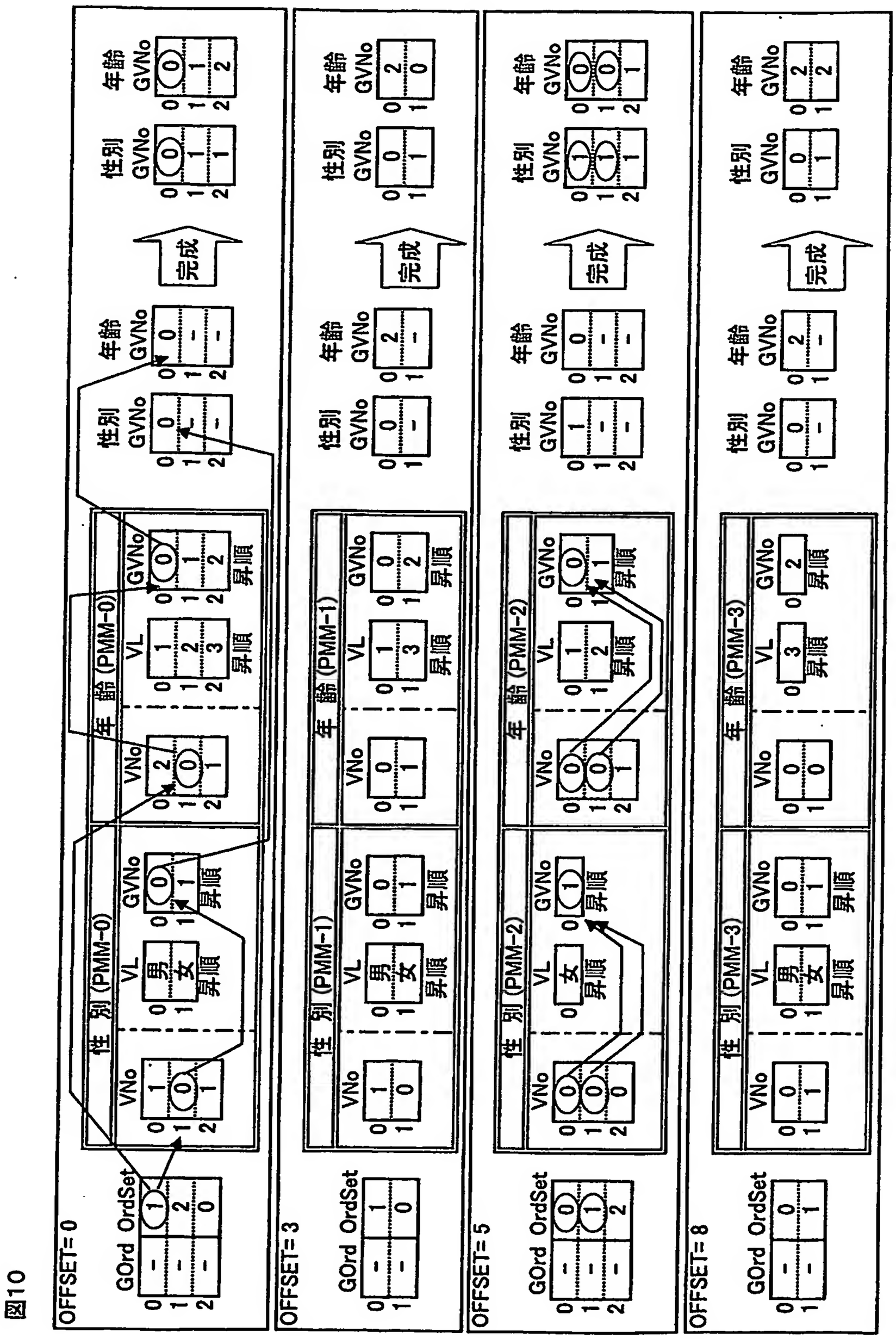
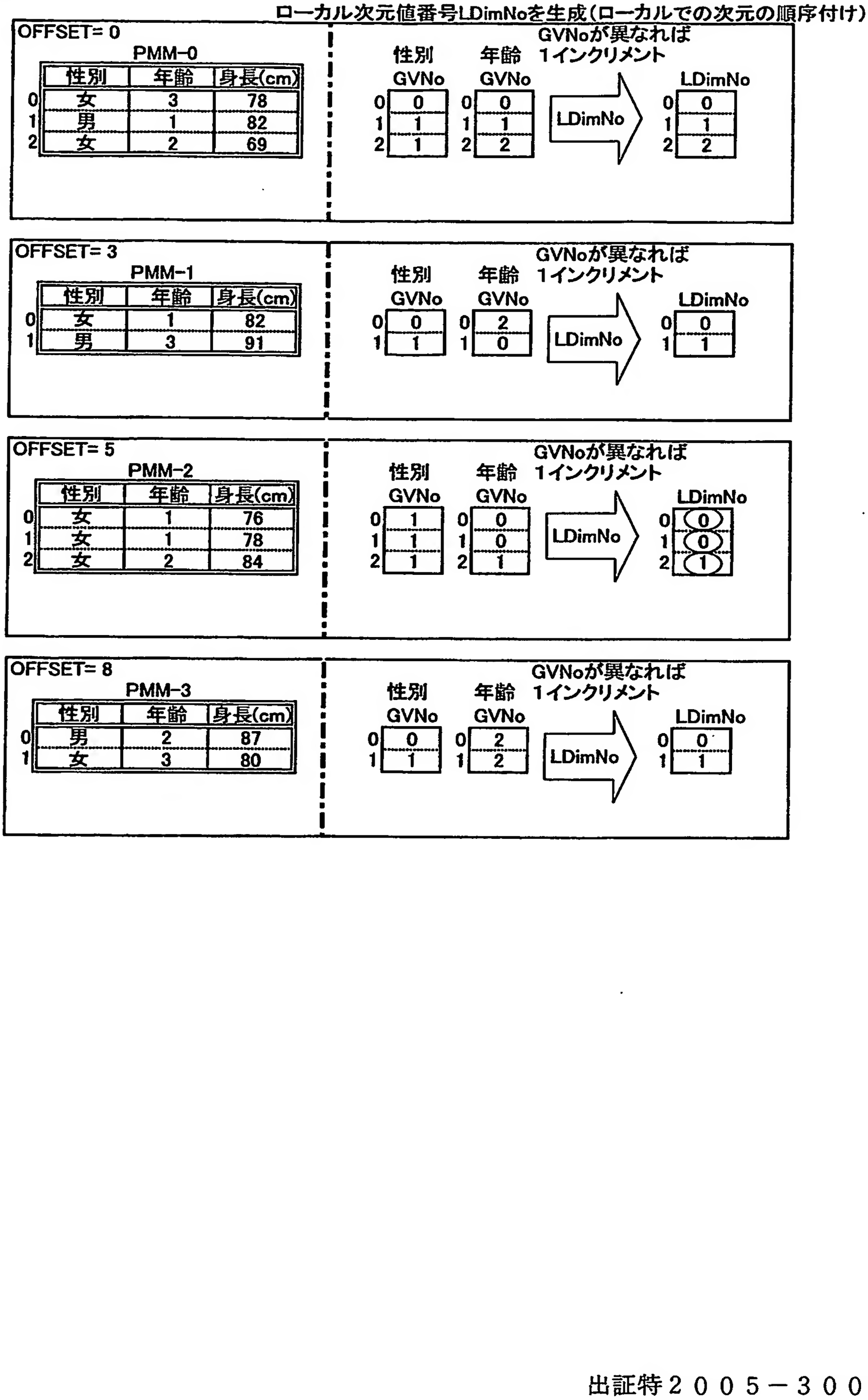

OFFSET= 0
OFFSET= 3
OFFSET= 5
OFFSET= 8
GOrd OrdSet
性別 (PMM-0)
性別 (PMM-1)
性別 (PMM-2)
性別 (PMM-3)
年齢 (PMM-0)
年齢 (PMM-1)
年齢 (PMM-2)
年齢 (PMM-3)
VL
VNo
GVNo
男
女
昇順
完成

図10

【図１１】

図11


ローカル次元値番号LDimNoを生成(ローカルでの次元の順序付け)
OFFSET= 0
PMM-0
性別 年齢 身長(cm)
0 女 3 78
1 男 1 82
2 女 2 69
GVNoが異なれば
1インクリメント
性別 GVNo
0 0
1 1
2 1
年齢 GVNo
0 0
1 1
2 2
LDimNo
LDimNo
0 0
1 1
2 2
OFFSET= 3
PMM-1
性別 年齢 身長(cm)
0 女 1 82
1 男 3 91
GVNoが異なれば
1インクリメント
性別 GVNo
0 0
1 1
年齢 GVNo
0 2
1 0
LDimNo
LDimNo
0 0
1 1
OFFSET= 5
PMM-2
性別 年齢 身長(cm)
0 女 1 76
1 女 1 78
2 女 2 84
GVNoが異なれば
1インクリメント
性別 GVNo
0 1
1 1
2 1
年齢 GVNo
0 0
1 0
2 1
LDimNo
LDimNo
0 0
1 0
2 1
OFFSET= 8
PMM-3
性別 年齢 身長(cm)
0 男 2 87
1 女 3 80
GVNoが異なれば
1インクリメント
性別 GVNo
0 0
1 1
年齢 GVNo
0 2
1 2
LDimNo
LDimNo
0 0
1 1

【図１２】

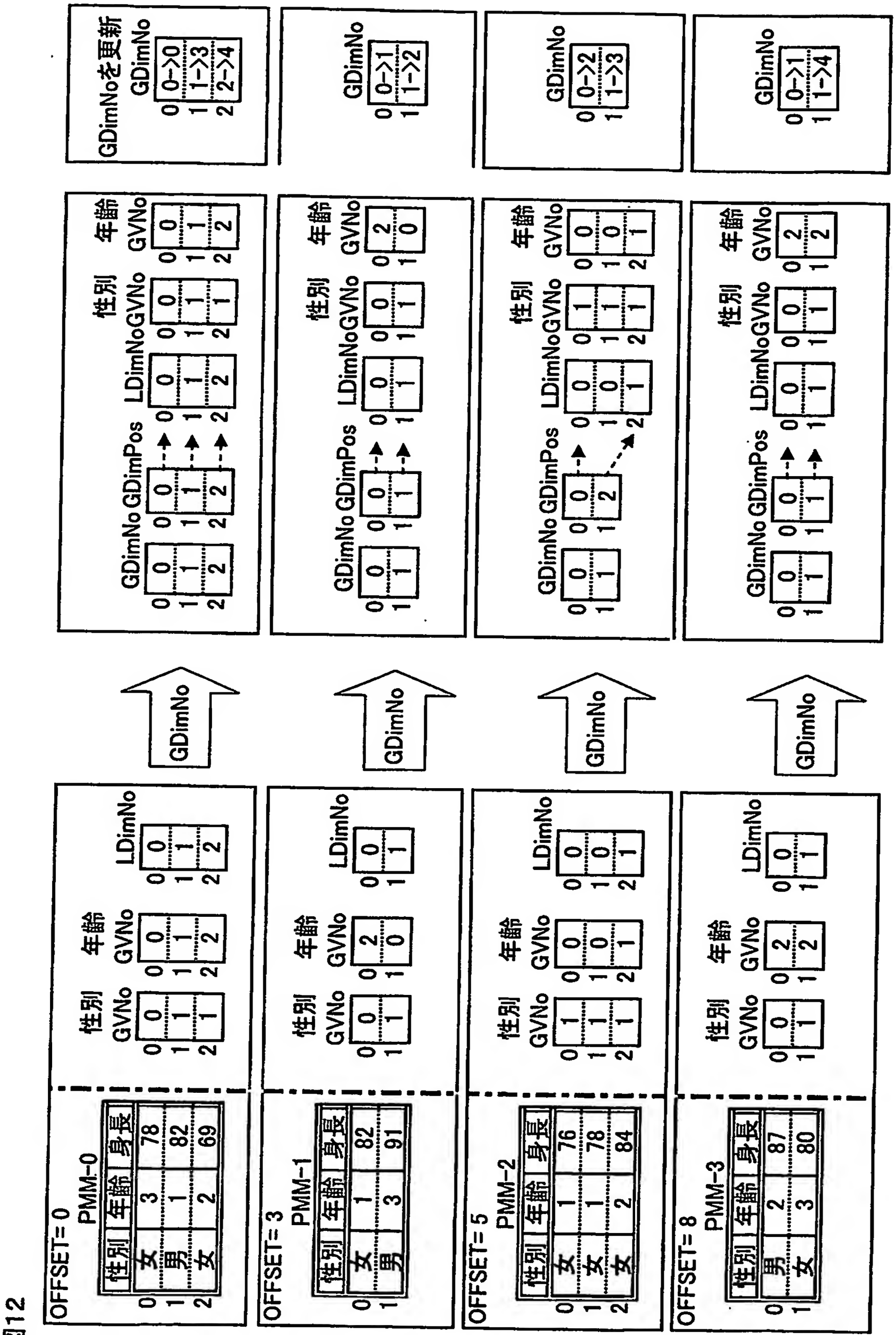

GDimNoを更新
GDimNo
0 0->0
1 1->3
2 2->4
GDimNo
0 0->1
1 1->2
GDimNo
0 0->2
1 1->3
GDimNo
0 0->1
1 1->4
GDimNo
GDimPos
LDimNo
LDimNoGVNo
性別
年齢
GVNo
GDimNo
LDimNo
OFFSET= 0
PMM-0
OFFSET= 3
PMM-1
OFFSET= 5
PMM-2
OFFSET= 8
PMM-3
性別
年齢
身長
女 3 78
男 1 82
女 2 69
女 1 82
男 3 91
女 1 76
女 1 78
女 2 84
男 2 87
女 3 80

図12

【図１３】

図13

	LDimNo	GNo1	GNo2	GDimNo
PMM-0	0	0	0	0
	1	1	1	3
	2	1	2	4
PMM-1	0	0	2	1
	1	1	0	2
PMM-2	0	1	0	2
	1	1	1	3
PMM-3	0	0	2	1
	1	1	1	2

【図１４】

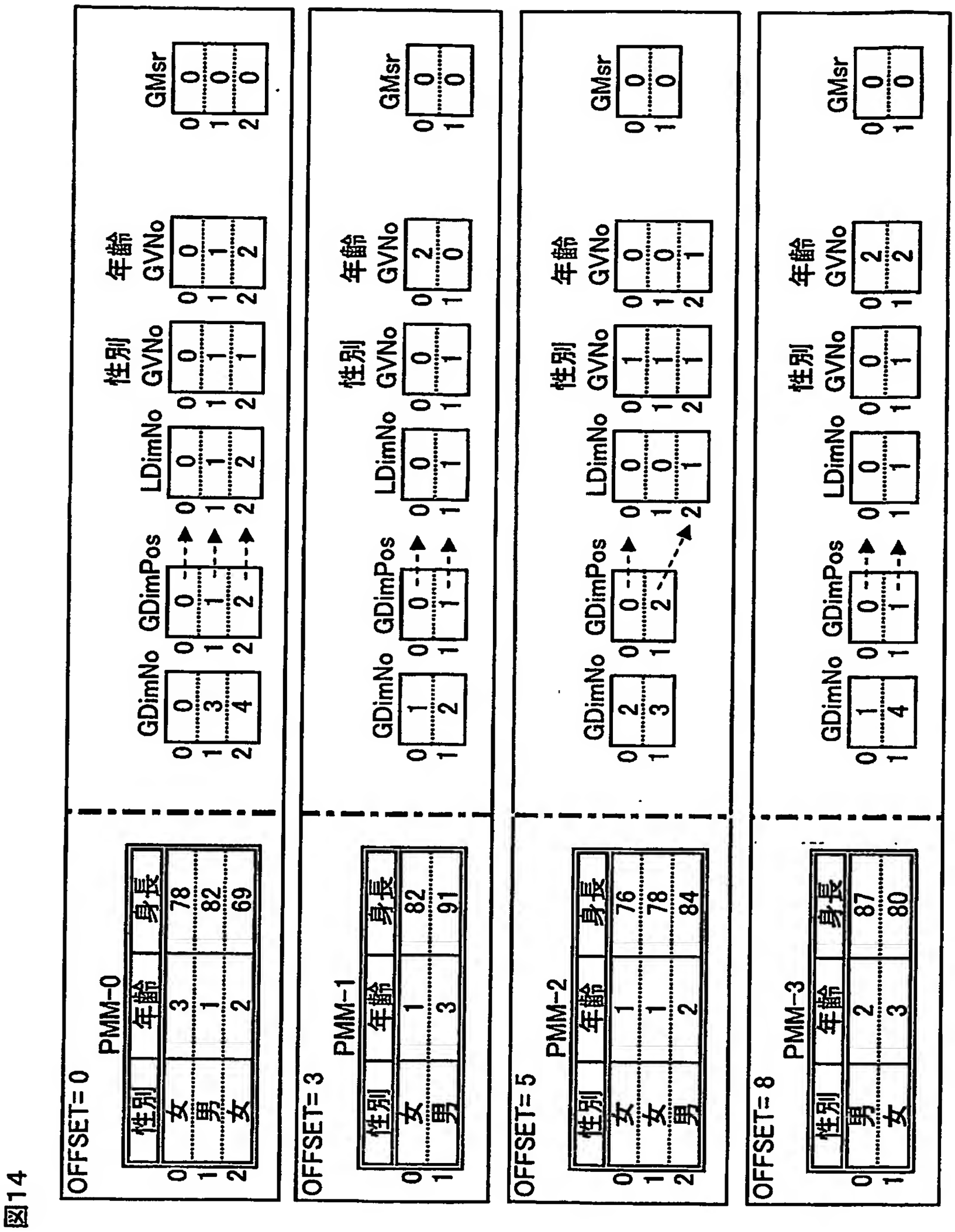
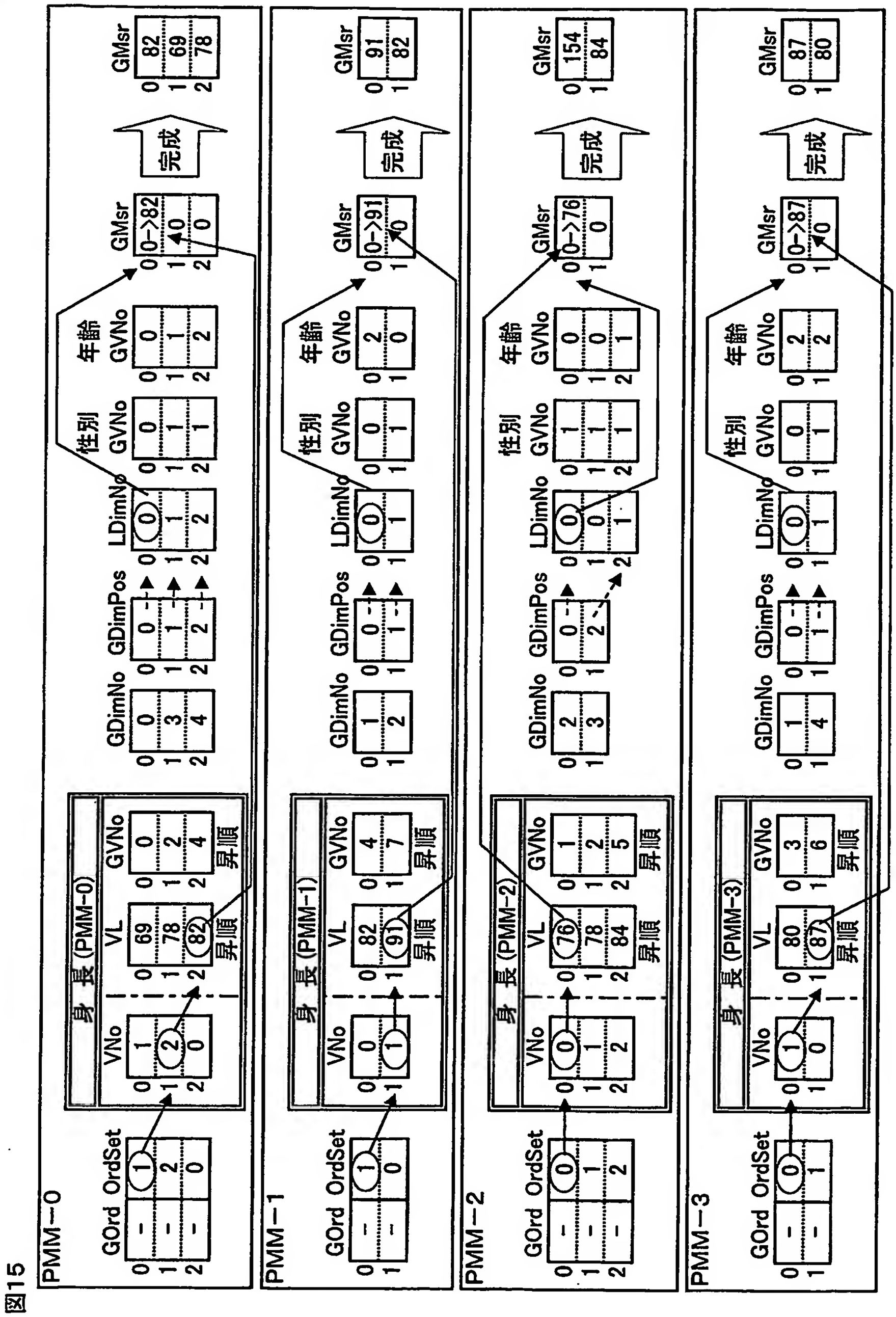

OFFSET=0
PMM-0
性別
女
男
女
年齢
3
1
2
身長
78
82
69
GVNo
0
1
2
GDimNo
0
3
4
GDimPos
0
1
2
LDimNo
0
1
2
GMsr
0
1
2
OFFSET=3
PMM-1
性別
女
男
年齢
1
3
身長
82
91
GVNo
0
2
1
GDimNo
1
2
GDimPos
0
1
LDimNo
0
1
GMsr
0
1
OFFSET=5
PMM-2
性別
女
女
男
年齢
1
1
2
身長
76
78
84
GVNo
0
1
2
GDimNo
2
3
GDimPos
0
2
1
LDimNo
0
1
2
GMsr
0
1
OFFSET=8
PMM-3
性別
男
女
年齢
2
3
身長
87
80
GVNo
0
2
1
2
GDimNo
1
4
GDimPos
0
1
1
LDimNo
0
0
1
GMsr
0
1

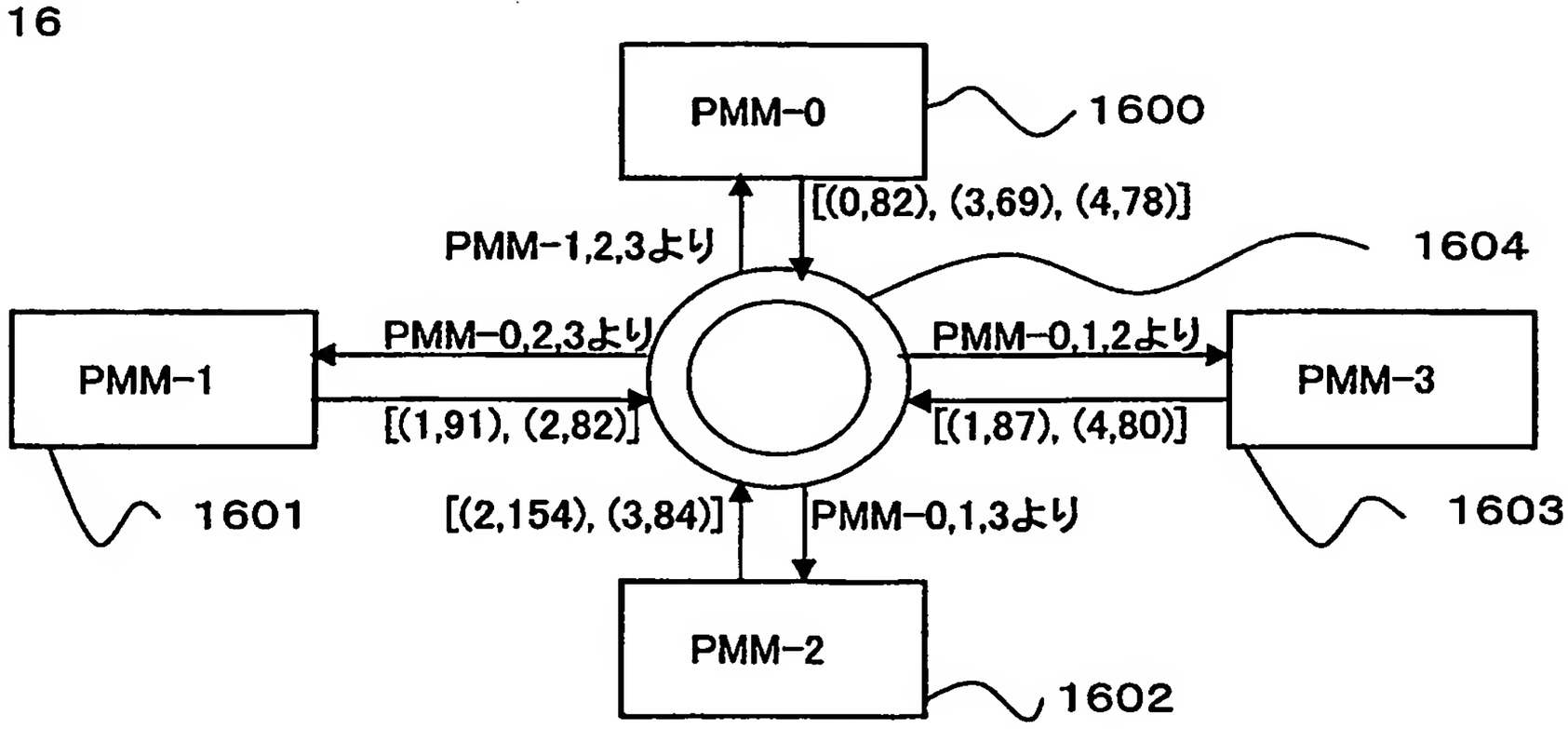
図14

【図１５】


PMM−0
PMM−1
PMM−2
PMM−3
GOrd OrdSet
VNo
長 (PMM−0)
長 (PMM−1)
長 (PMM−2)
長 (PMM−3)
VL
GVNo
昇順
GDimNo
GDimPos
LDimNo
性別
年齢
GMsr
0−>82
0−>91
0−>76
0−>87
完成
82
69
78
91
76
84
80
87
154
84
図15

【図１６】

図16


PMM-0
1600
[(0,82), (3,69), (4,78)]
PMM-1,2,3より
1604
PMM-0,2,3より
PMM-0,1,2より
PMM-1
PMM-3
[(1,91), (2,82)]
[(1,87), (4,80)]
1601
[(2,154), (3,84)]
PMM-0,1,3より
1603
PMM-2
1602

【図１７】

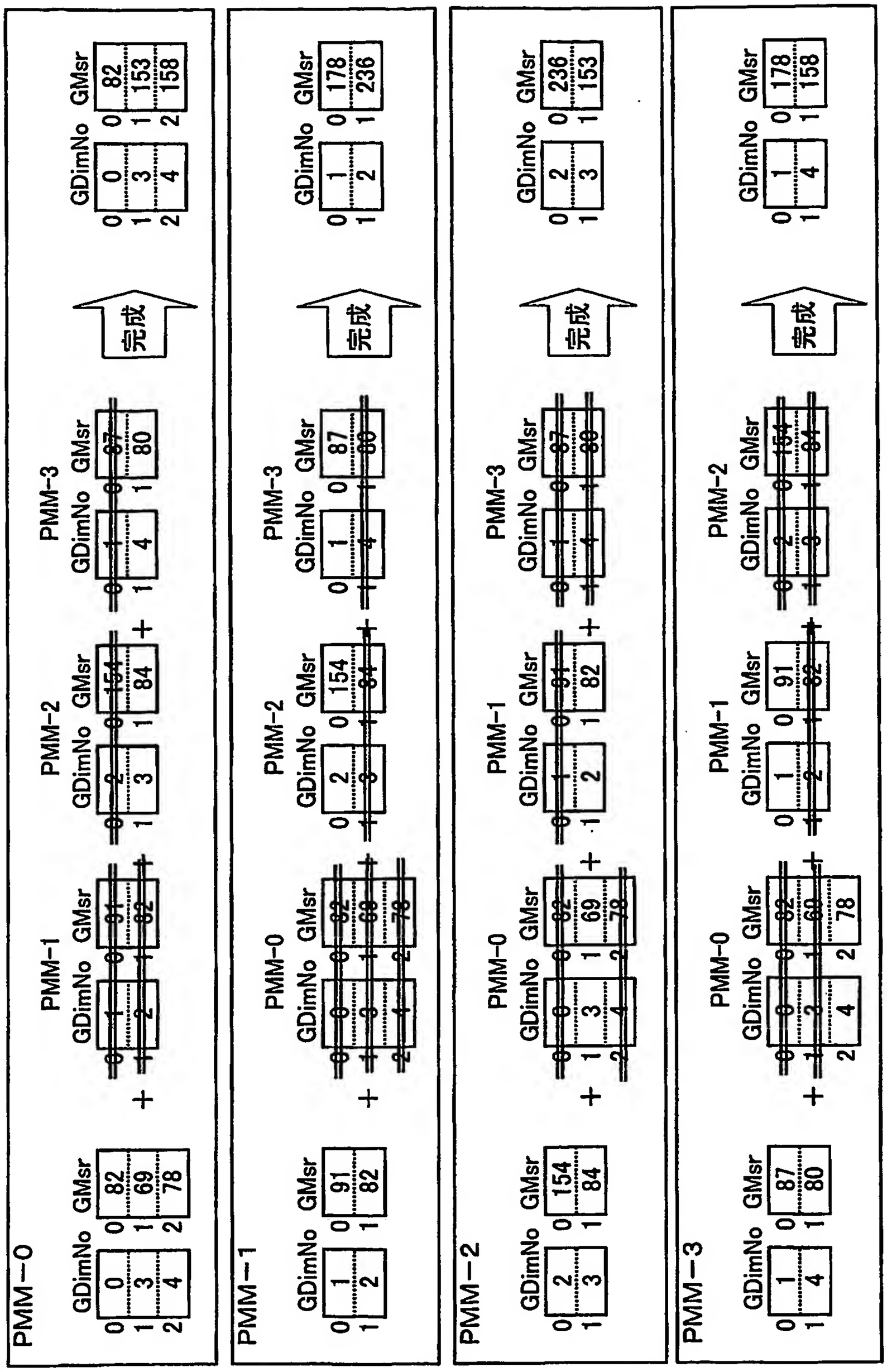
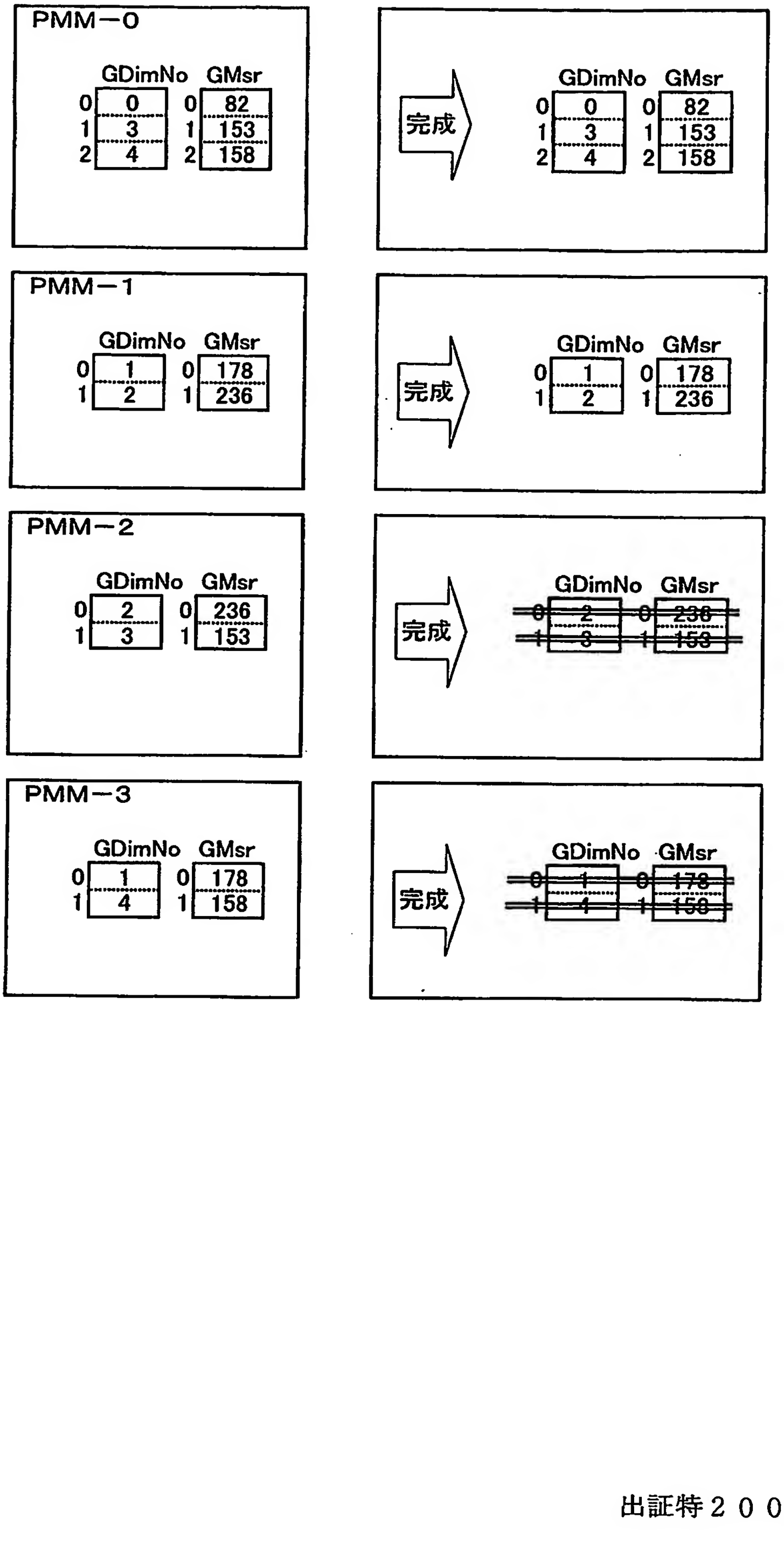


図17
PMM−0
GDimNo
GMsr
0
3
4
82
69
78
PMM-1
1
2
91
82
PMM-2
2
3
154
84
PMM-3
1
4
87
80
完成
0
3
4
82
153
158
PMM−1
1
2
91
82
PMM-0
0
3
4
82
69
78
PMM-2
2
3
154
84
PMM-3
1
4
87
80
完成
1
2
178
236
PMM−2
2
3
154
84
PMM-0
0
3
4
82
69
78
PMM-1
1
2
91
82
PMM-3
1
4
87
80
完成
2
3
236
153
PMM−3
1
4
87
80
PMM-0
0
3
4
82
69
78
PMM-1
1
2
91
82
PMM-2
2
3
154
84
完成
1
4
178
158

【図１８】

図18


PMM−0
GDimNo GMsr
0 0
1 3
2 4
0 82
1 153
2 158
完成
GDimNo GMsr
0 0
1 3
2 4
0 82
1 153
2 158
PMM−1
GDimNo GMsr
0 1
1 2
0 178
1 236
完成
GDimNo GMsr
0 1
1 2
0 178
1 236
PMM−2
GDimNo GMsr
0 2
1 3
0 236
1 153
完成
GDimNo GMsr
0 2
1 3
0 236
1 153
PMM−3
GDimNo GMsr
0 1
1 4
0 178
1 158
完成
GDimNo GMsr
0 1
1 4
0 178
1 158

【図１９】

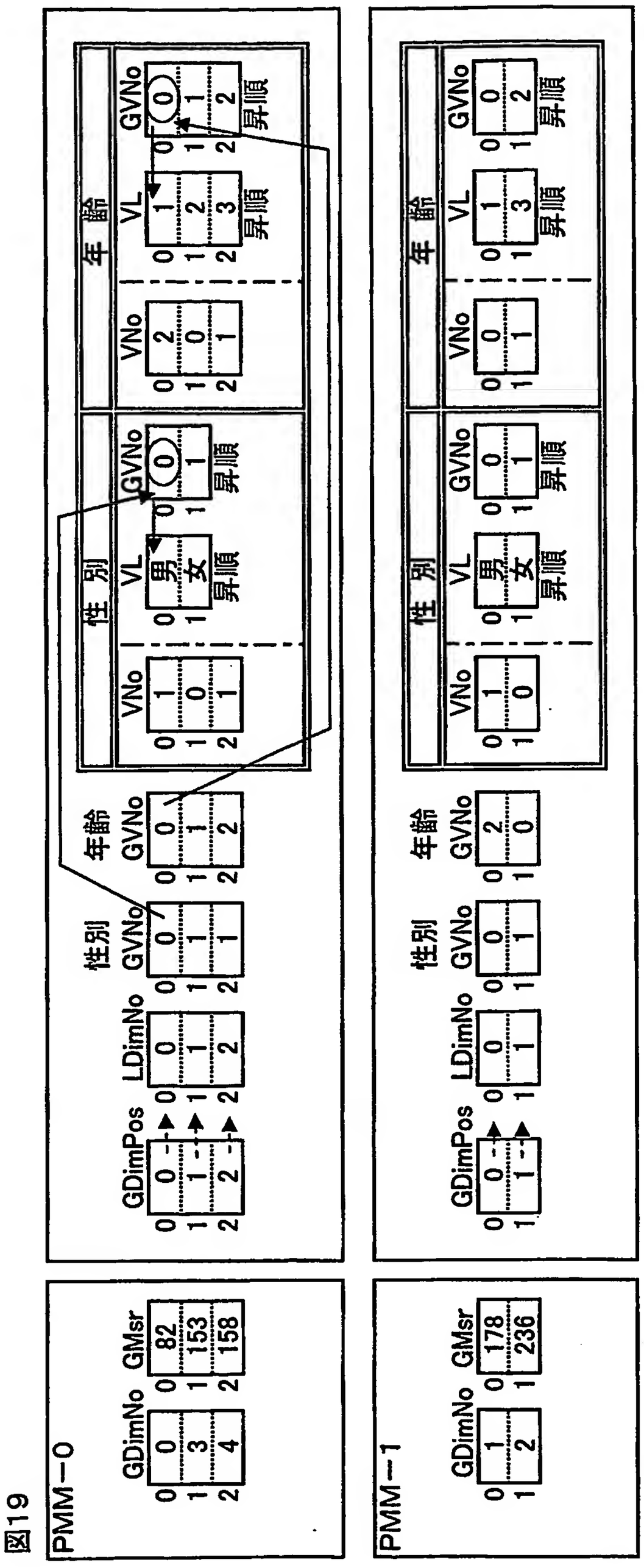
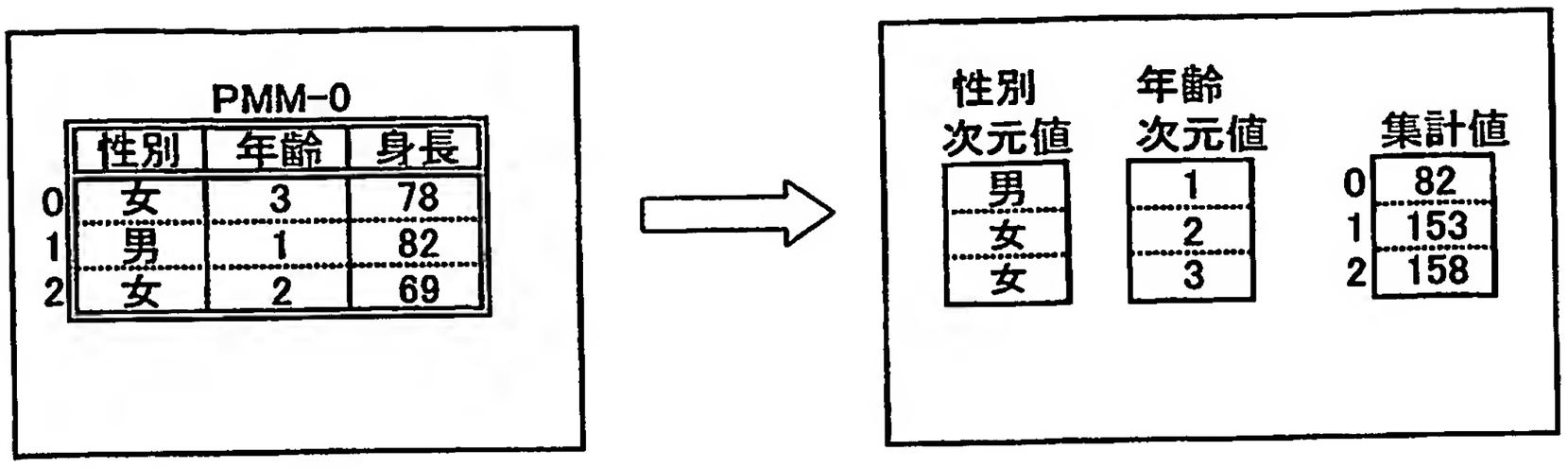


図19

【図２０】

図20


PMM-0
性別 年齢 身長
0 女 3 78
1 男 1 82
2 女 2 69
性別 次元値
男
女
女
年齢 次元値
1
2
3
集計値
0 82
1 153
2 158
PMM-1
性別 年齢 身長
0 女 1 82
1 男 3 91
性別 次元値
男
女
年齢 次元値
3
1
集計値
0 178
1 236
PMM-2
性別 年齢 身長
0 女 1 76
1 女 1 78
2 女 2 84
PMM-3
性別 年齢 身長
0 男 3 87
1 女 3 80

【図２１】

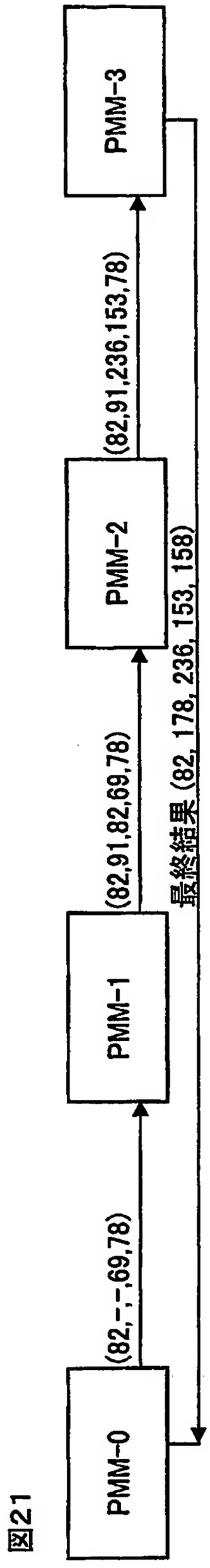
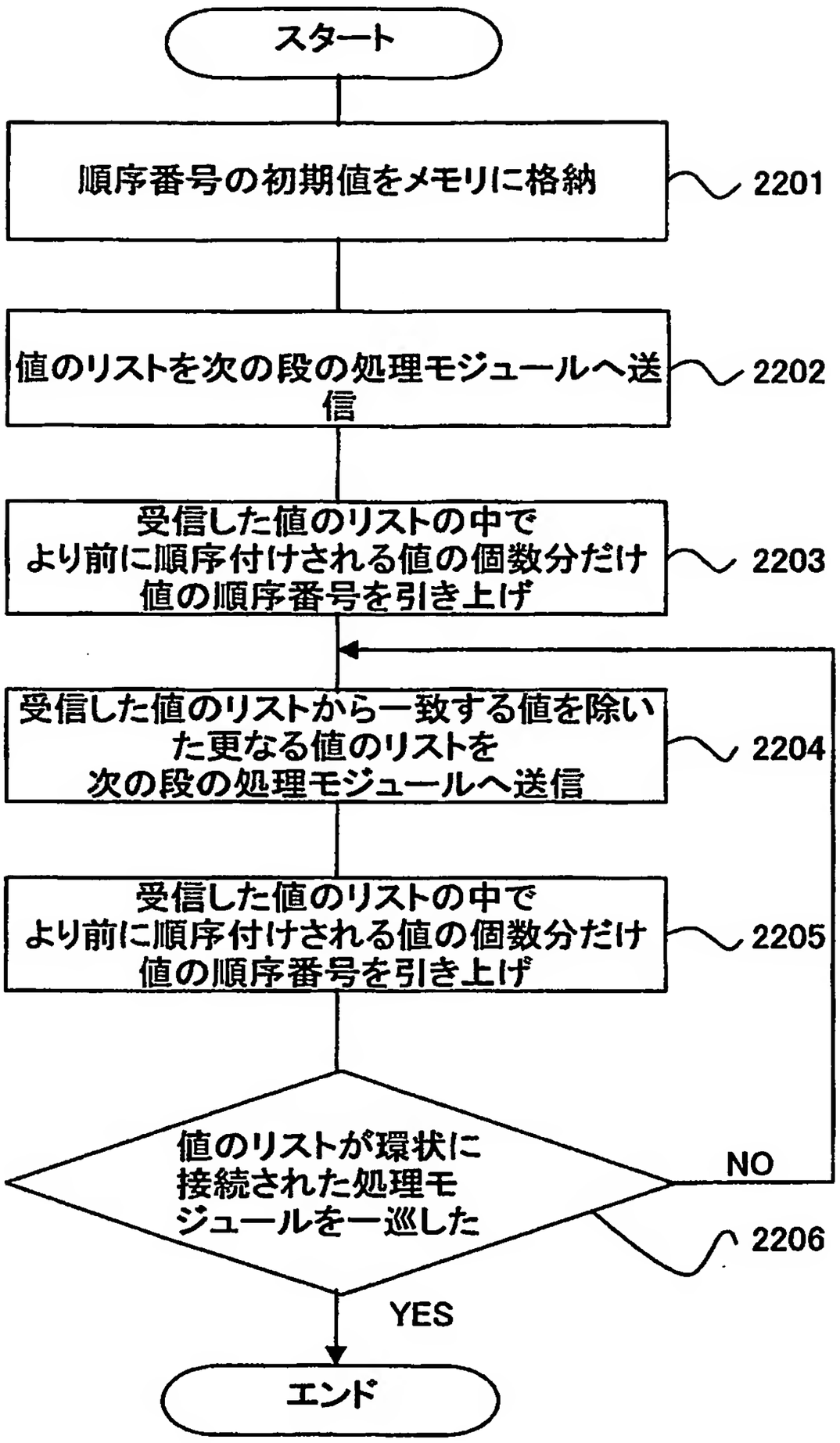

PMM-3
(82,91,236,153,78)
PMM-2
(82,9
(82,91,82,69,78)
最終結果 (82, 178, 236, 153, 158)
PMM-1
(82,-,-,69,78)
PMM-0

図21

【図２２】

図22


スタート
順序番号の初期値をメモリに格納
2201
値のリストを次の段の処理モジュールへ送信
2202
受信した値のリストの中で
より前に順序付けされる値の個数分だけ
値の順序番号を引き上げ
2203
受信した値のリストから一致する値を除いた更なる値のリストを
次の段の処理モジュールへ送信
2204
受信した値のリストの中で
より前に順序付けされる値の個数分だけ
値の順序番号を引き上げ
2205
値のリストが環状に
接続された処理モジュールを一巡した
NO
2206
YES
エンド

【図２３】

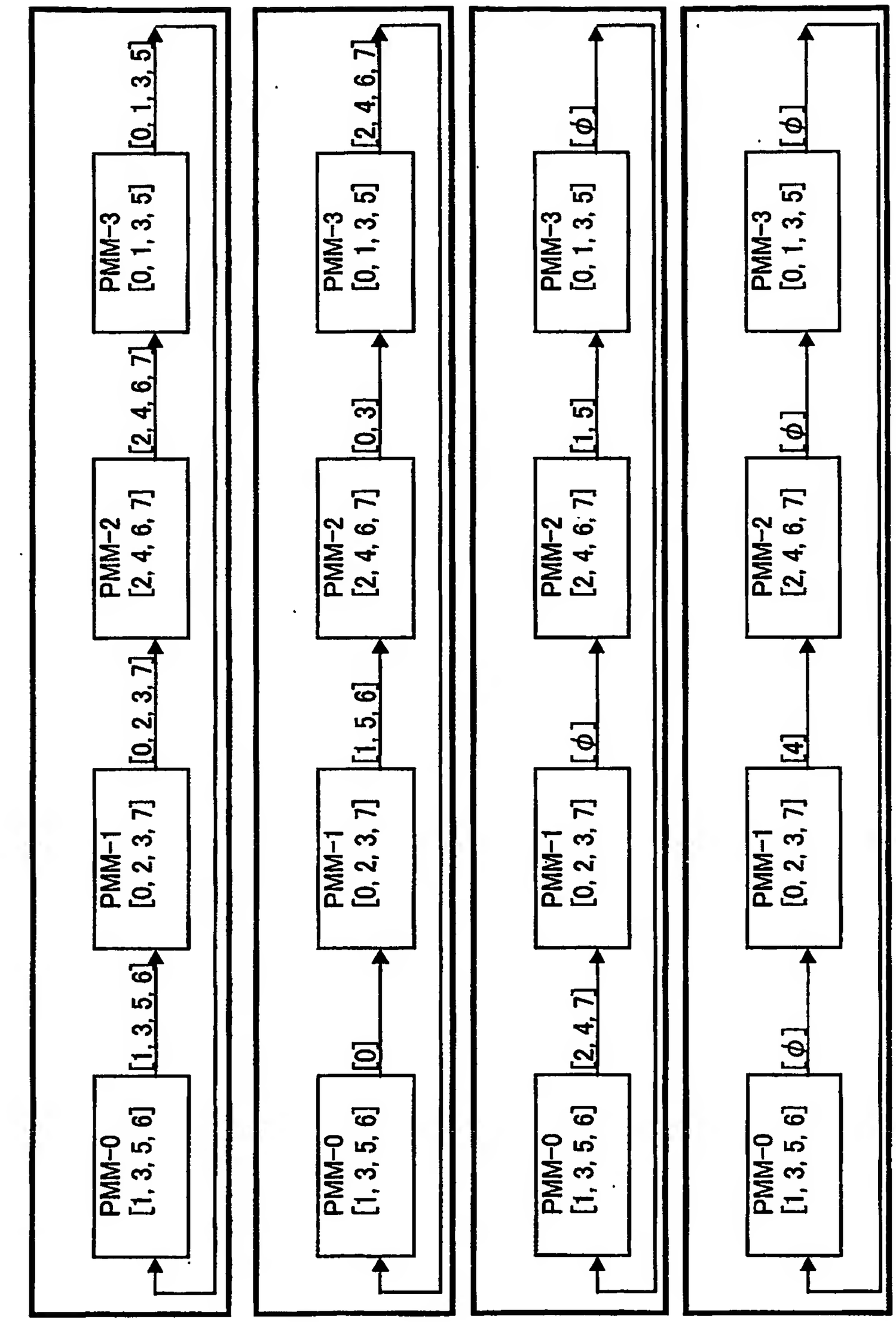

PMM-3 [0, 1, 3, 5]
[0, 1, 3, 5]
[2, 4, 6, 7]
PMM-2 [2, 4, 6, 7]
[0, 2, 3, 7]
PMM-1 [0, 2, 3, 7]
[1, 3, 5, 6]
PMM-0 [1, 3, 5, 6]
[2, 4, 6, 7]
[0, 3]
[1, 5, 6]
[0]
[1, 5]
[ϕ]
[2, 4, 7]
[4]
(A) ステップ1
(B) ステップ2
(C) ステップ3
(D) ステップ4

図23

図24

(A) ステップ1

受信済みリスト			
PMM-0	PMM-1	PMM-2	PMM-3
[0, 1, 3, 5]	[1, 3, 5, 6]	[0, 2, 3, 7]	[2, 4, 6, 7]

(B) ステップ2

受信済みリスト			
PMM-0	PMM-1	PMM-2	PMM-3
[0, 1, 3, 5]	[1, 3, 5, 6]	[0, 2, 3, 7]	[2, 4, 6, 7]
[2, 4, 6, 7]	[0]	[1, 5, 6]	[0, 3]

(C) ステップ3

受信済みリスト			
PMM-0	PMM-1	PMM-2	PMM-3
[0, 1, 3, 5]	[1, 3, 5, 6]	[0, 2, 3, 7]	[2, 4, 6, 7]
[2, 4, 6, 7]	[0]	[1, 5, 6]	[0, 3]
[φ]	[2, 4, 7]	[φ]	[1, 5]

(D)ステップ4

受信済みリスト			
PMM-0	PMM-1	PMM-2	PMM-3
[0, 1, 3, 5]	[1, 3, 5, 6]	[0, 2, 3, 7]	[2, 4, 6, 7]
[2, 4, 6, 7]	[0]	[1, 5, 6]	[0, 3]
[φ]	[2, 4, 7]	[φ]	[1, 5]
[φ]	[φ]	[4]	[φ]

【図２５】

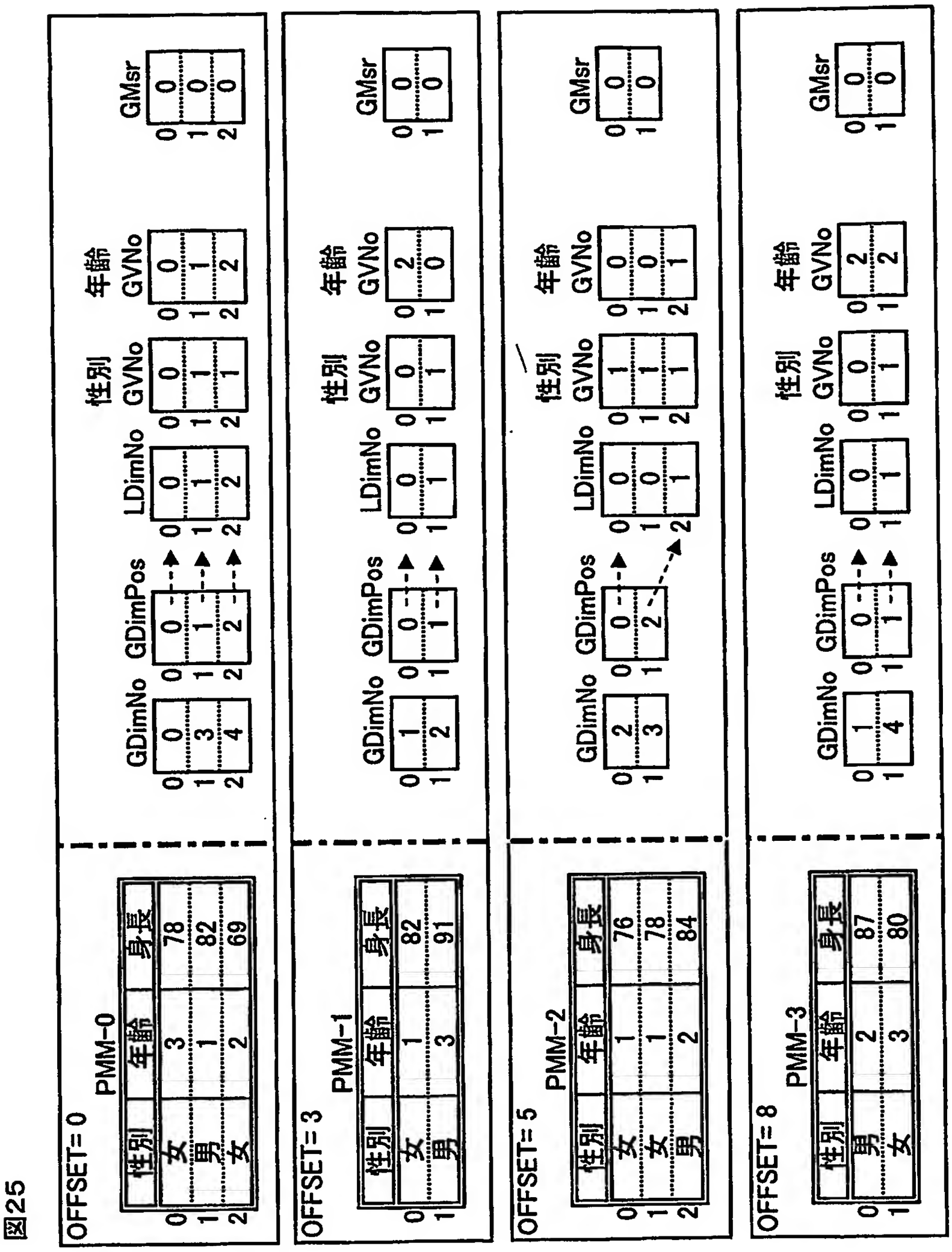
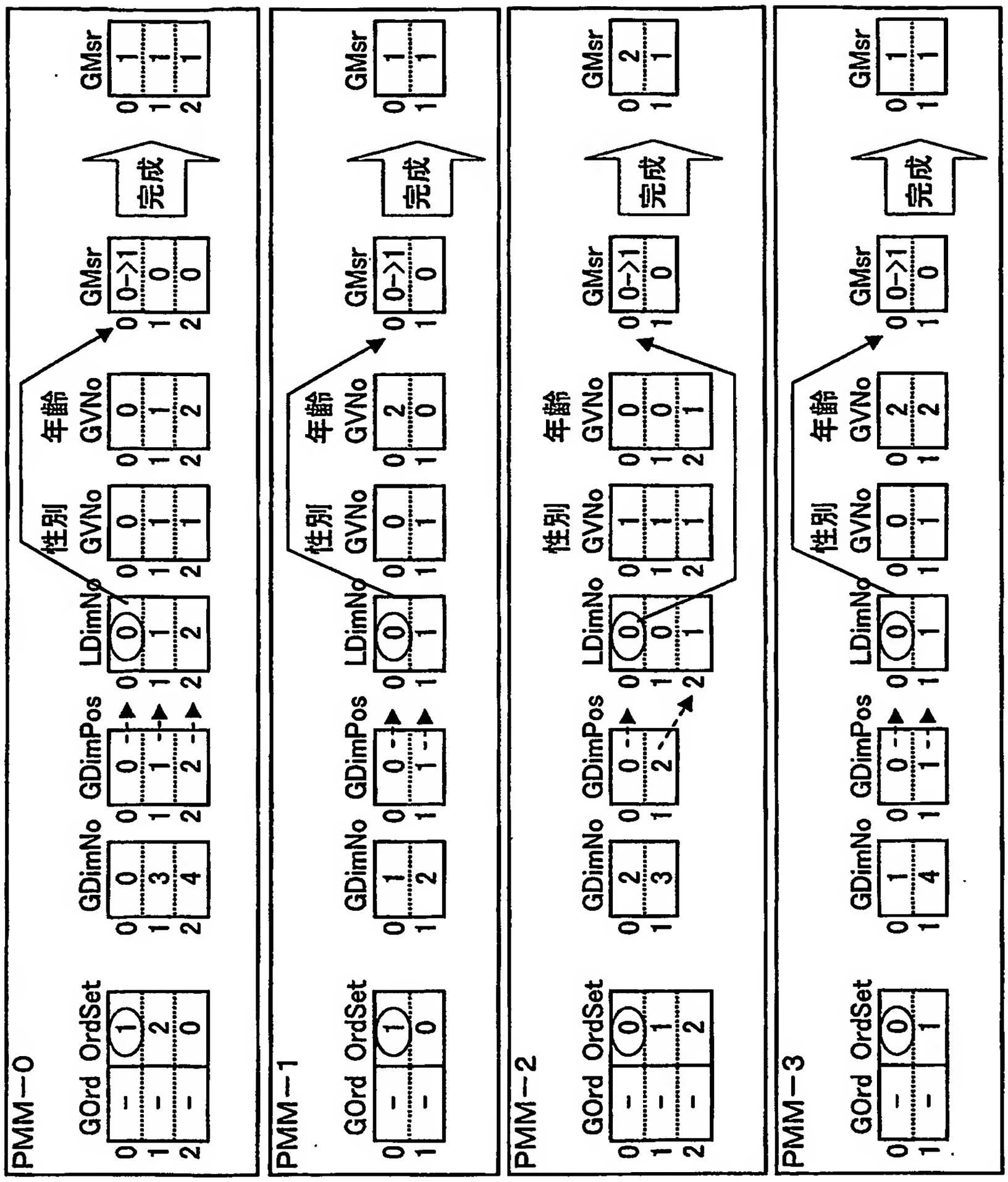

OFFSET=0
PMM-0
性別
女
男
女
年齢
3
1
2
身長
78
82
69
GVNo
0
1
2
GDimNo
0
3
4
GDimPos
0
1
2
LDimNo
0
1
2
GMsr
0
1
2
OFFSET=3
PMM-1
性別
女
男
年齢
1
3
身長
82
91
GVNo
0
2
0
GDimNo
1
2
GDimPos
0
1
LDimNo
0
1
GMsr
0
1
OFFSET=5
PMM-2
性別
女
女
男
年齢
1
1
2
身長
76
78
84
GVNo
0
1
2
GDimNo
2
3
GDimPos
0
1
2
LDimNo
0
1
2
GMsr
0
1
OFFSET=8
PMM-3
性別
男
女
年齢
2
3
身長
87
80
GVNo
0
2
1
2
GDimNo
1
4
GDimPos
0
1
LDimNo
0
1
GMsr
0
1

図25

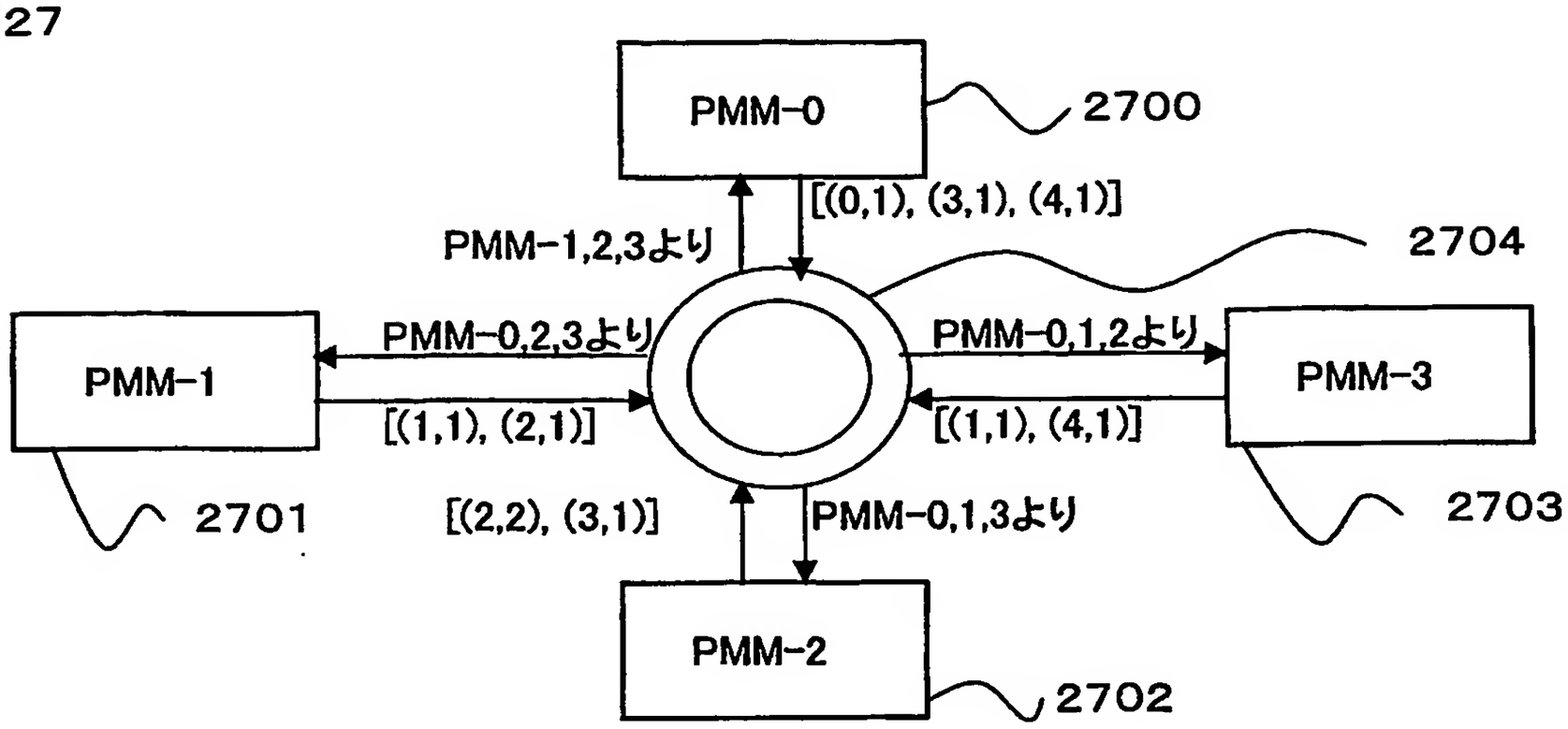
【図２６】

図26


PMM−0
PMM−1
PMM−2
PMM−3
GOrd
OrdSet
GDimNo
GDimPos
LDimNo
性別
GVNo
年齢
GMsr
0−>1
完成

【図２７】

図27


PMM-0
2700
[(0,1), (3,1), (4,1)]
PMM-1,2,3より
2704
PMM-0,2,3より
PMM-0,1,2より
PMM-1
PMM-3
[(1,1), (2,1)]
[(1,1), (4,1)]
2701
[(2,2), (3,1)]
PMM-0,1,3より
2703
PMM-2
2702

【図２８】

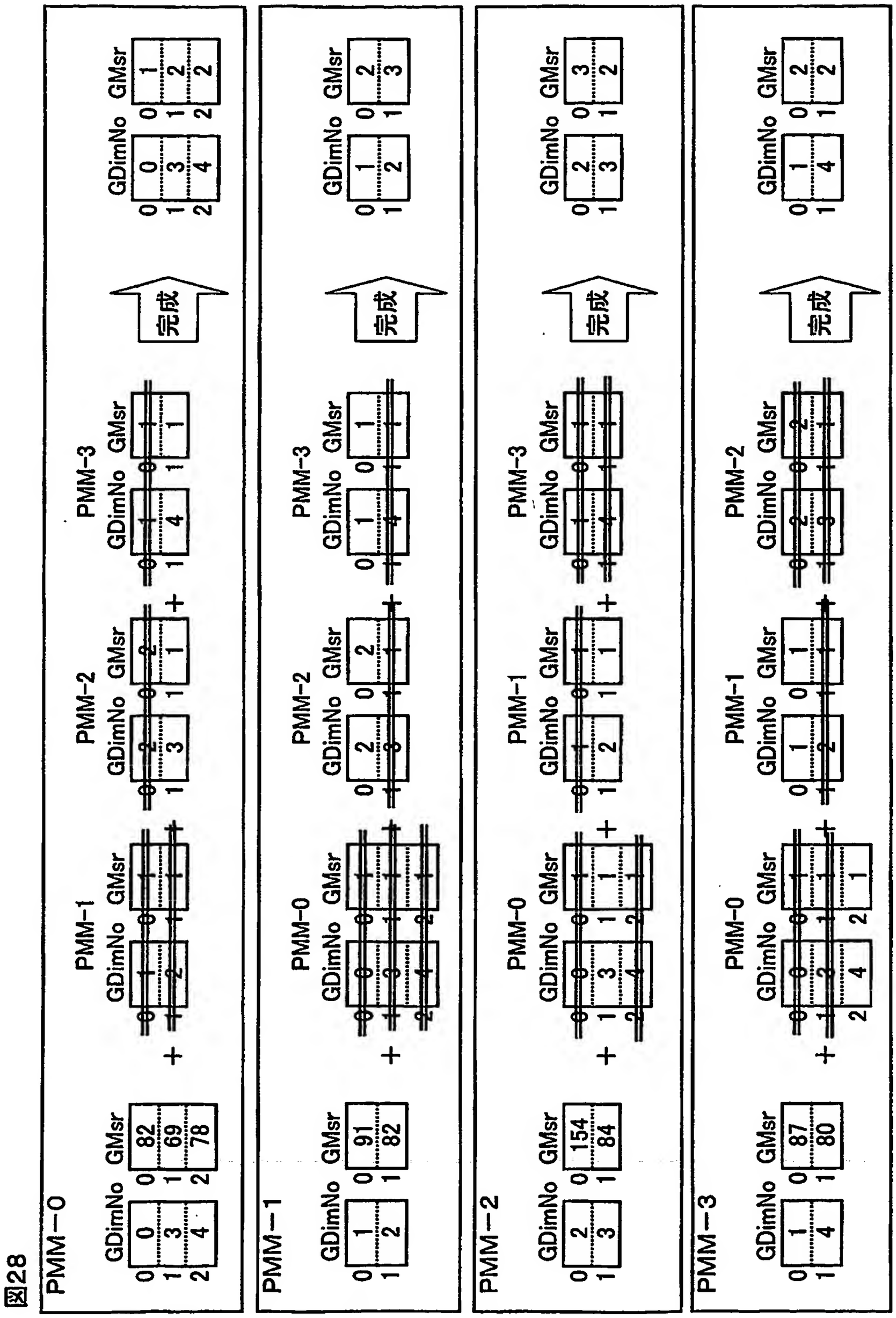
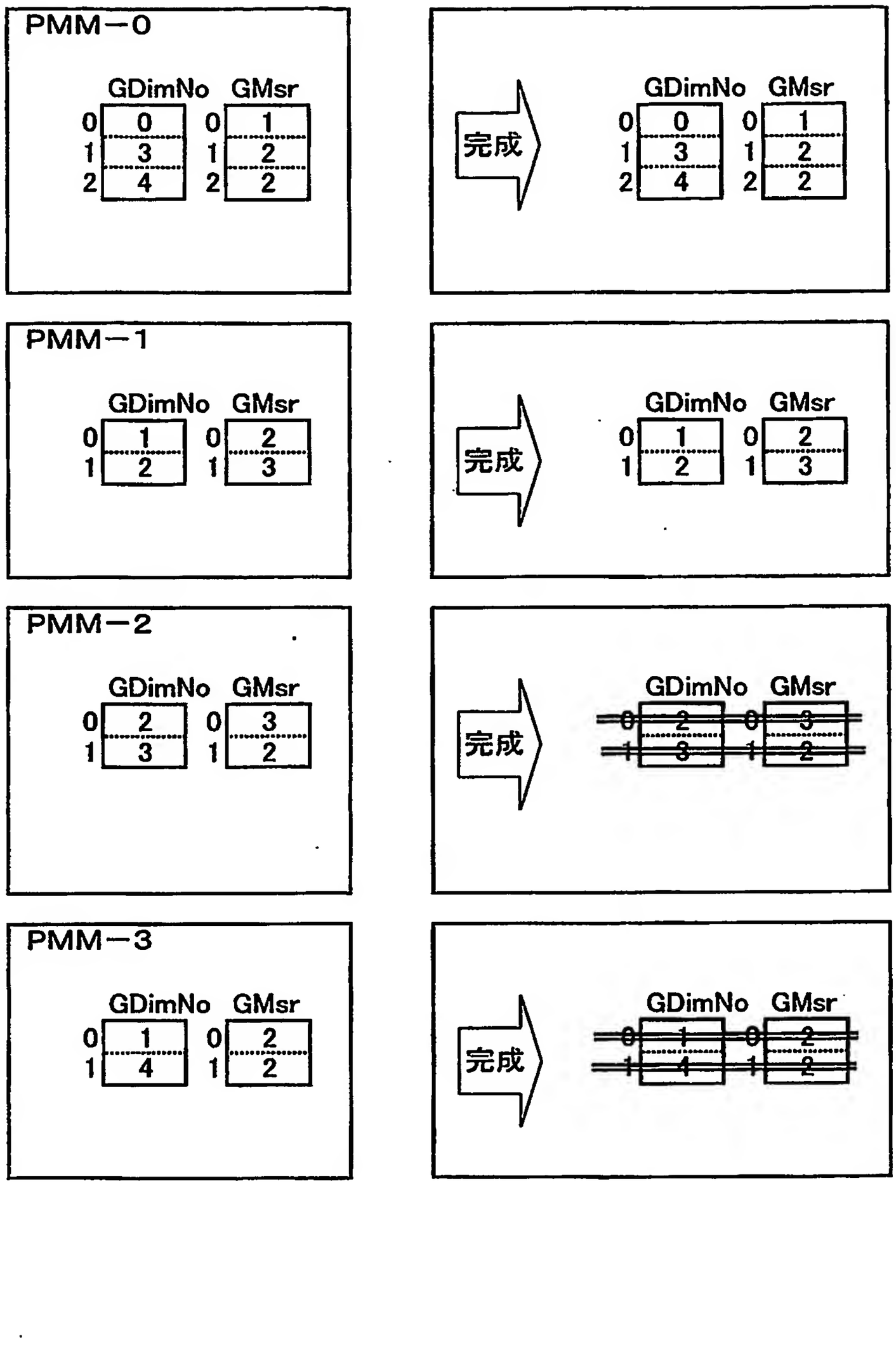


図28
PMM−0
PMM−1
PMM−2
PMM−3
GDimNo
GMsr
0 1 2
0 3 4
82 69 78
91 82
1 2
2 3
154 84
1 4
87 80
+
PMM-0
PMM-1
PMM-2
PMM-3
完成
1 2 2
2 3
3 2
2 2

【図２９】

図29


PMM−0
GDimNo GMsr
0 0 0 1
1 3 1 2
2 4 2 2
完成
GDimNo GMsr
0 0 0 1
1 3 1 2
2 4 2 2
PMM−1
GDimNo GMsr
0 1 0 2
1 2 1 3
完成
GDimNo GMsr
0 1 0 2
1 2 1 3
PMM−2
GDimNo GMsr
0 2 0 3
1 3 1 2
完成
GDimNo GMsr
PMM−3
GDimNo GMsr
0 1 0 2
1 4 1 2
完成
GDimNo GMsr

【図３０】

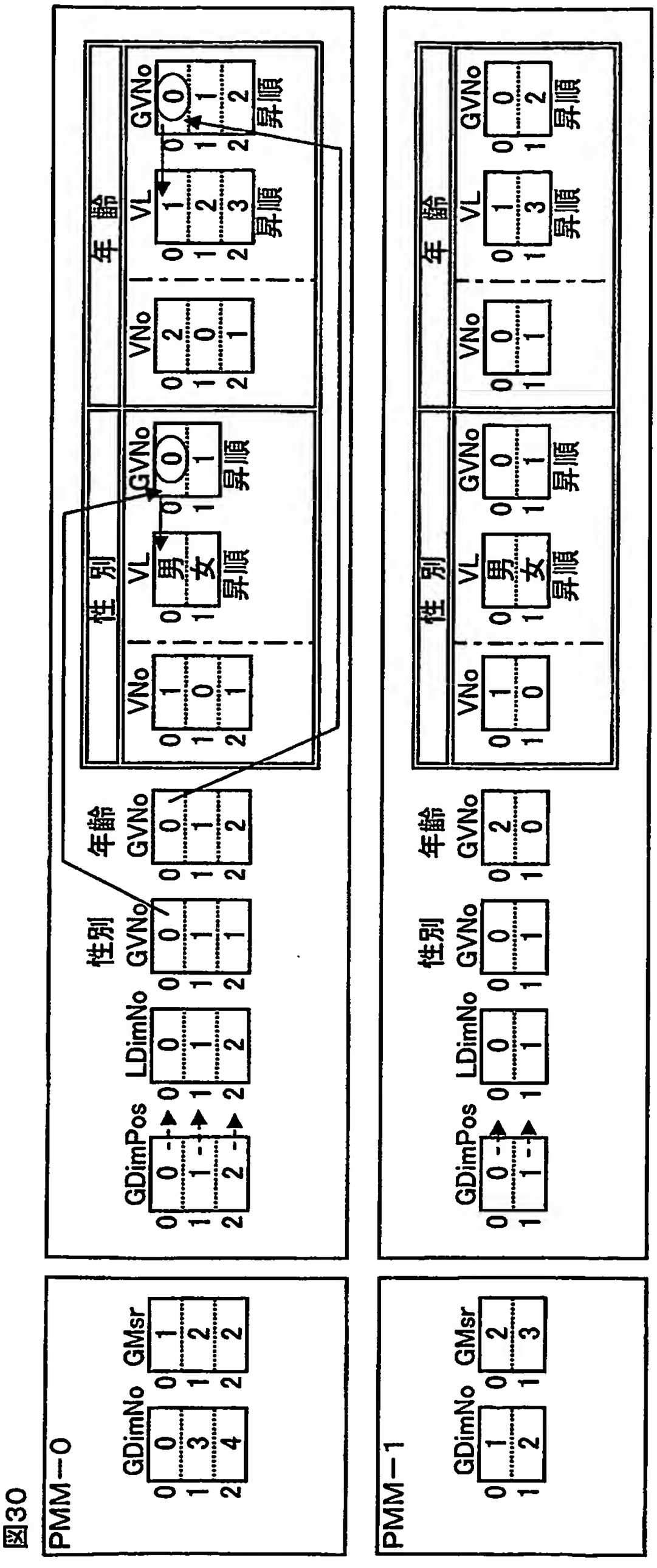
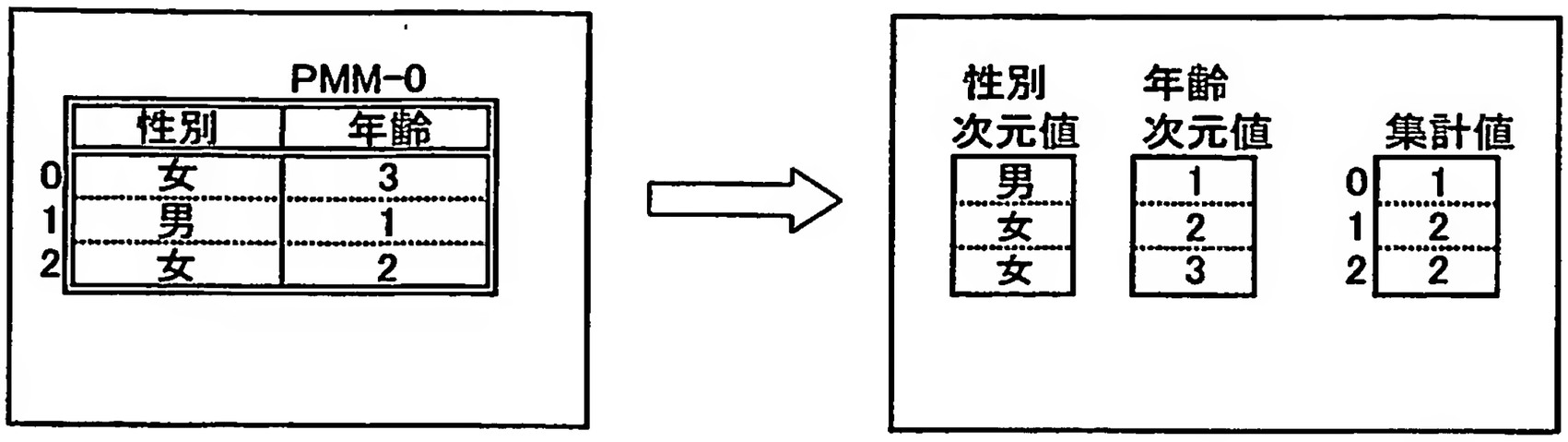


図30

【図３１】

図31


PMM-0
性別
年齢
0
女
3
1
男
1
2
女
2
性別
次元値
年齢
次元値
集計値
男
女
女
1
2
3
0
1
2
1
2
2
PMM-1
性別
年齢
0
女
1
1
男
3
性別
次元値
年齢
次元値
集計値
男
女
3
1
0
1
2
3
PMM-2
性別
年齢
0
女
1
1
女
1
2
女
2
PMM-3
性別
年齢
0
男
3
1
女
3

【図３２】

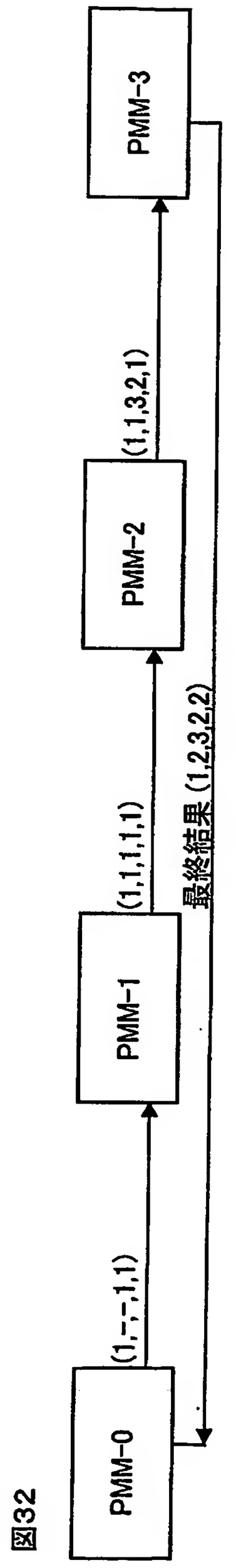


図32

【図３３】

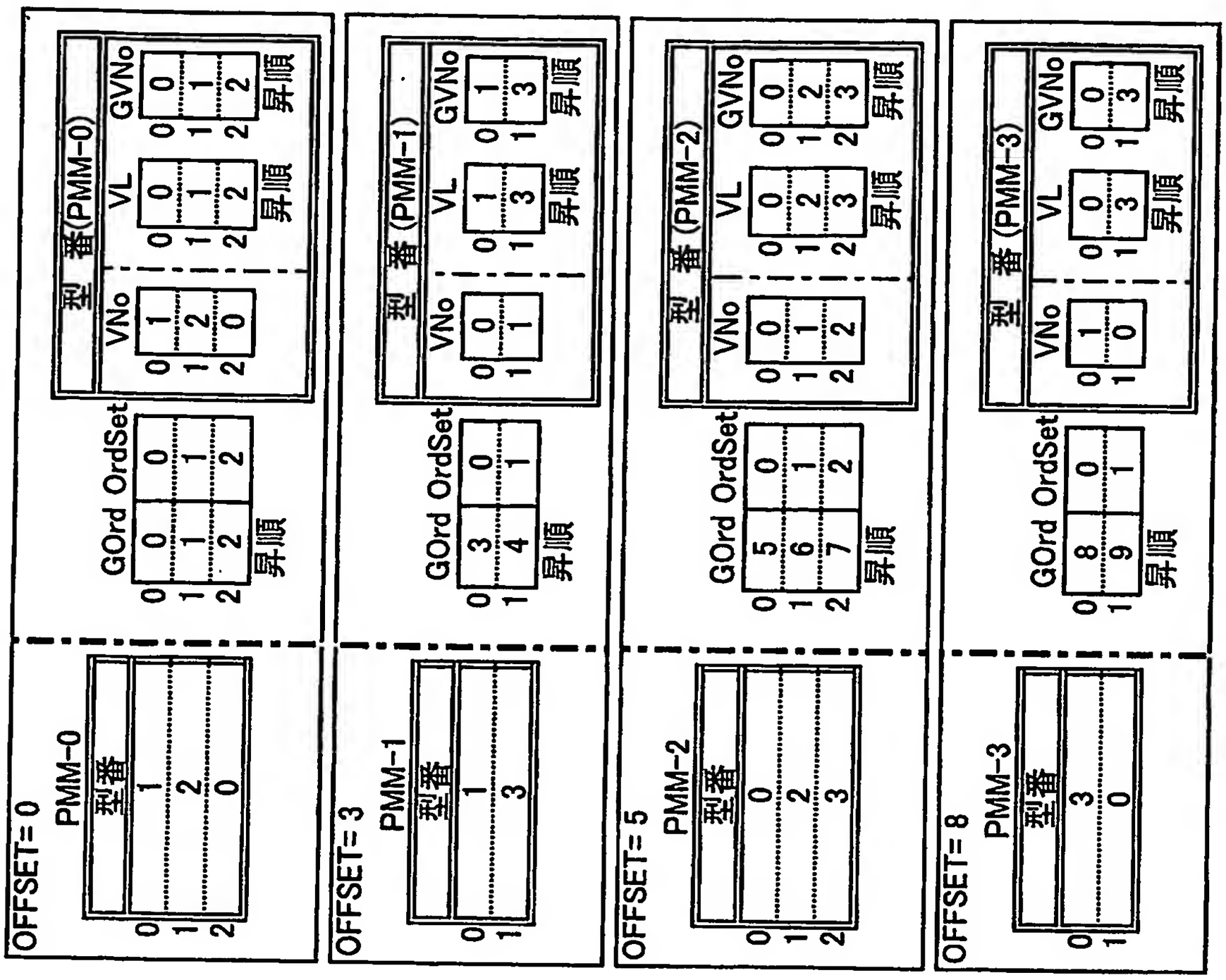
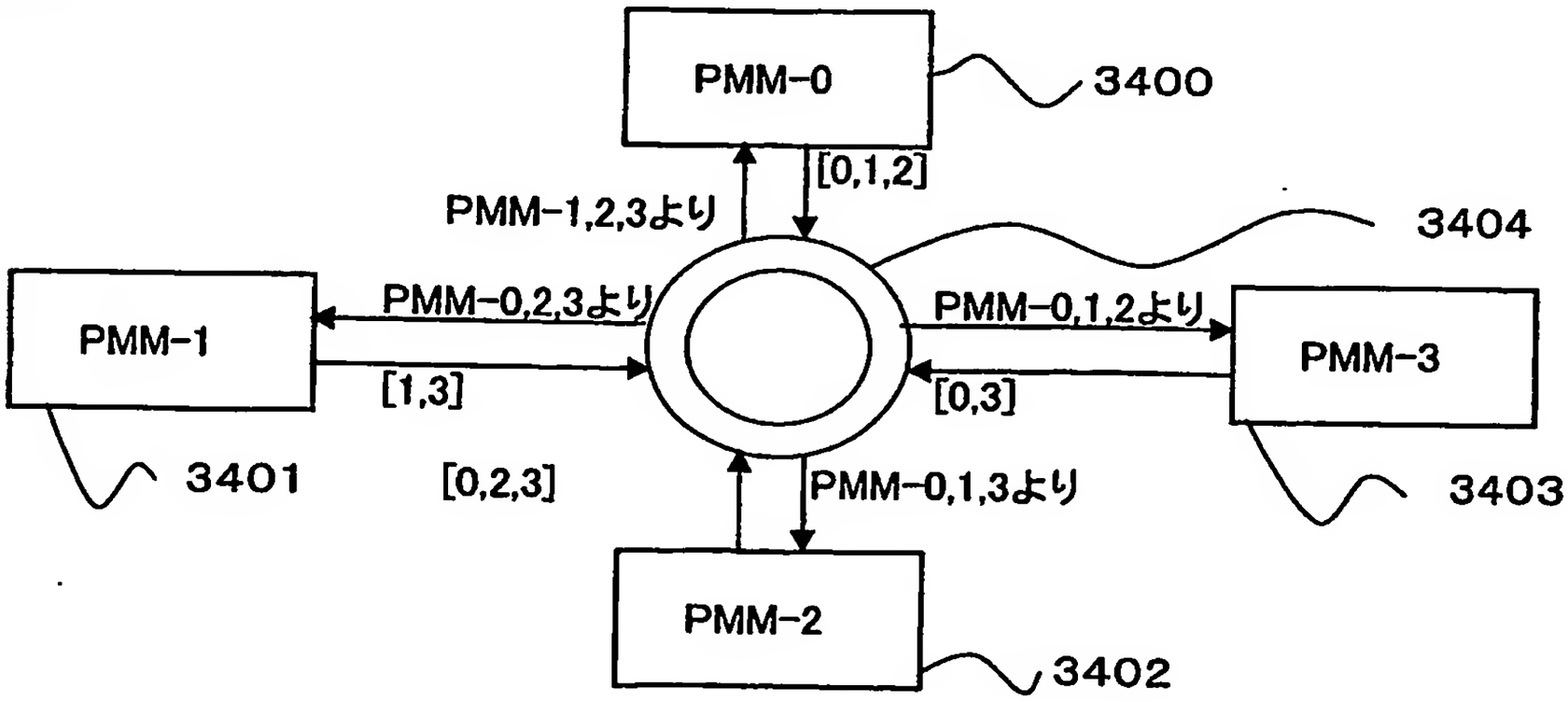


図33

【図３４】

図34



【図３５】

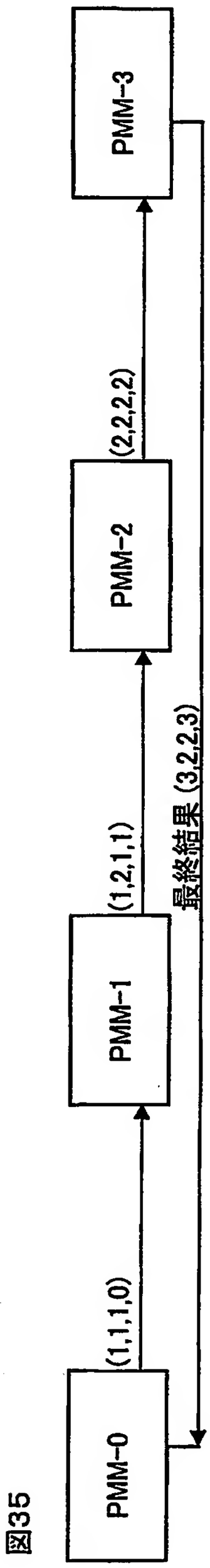


図35

【図３６】

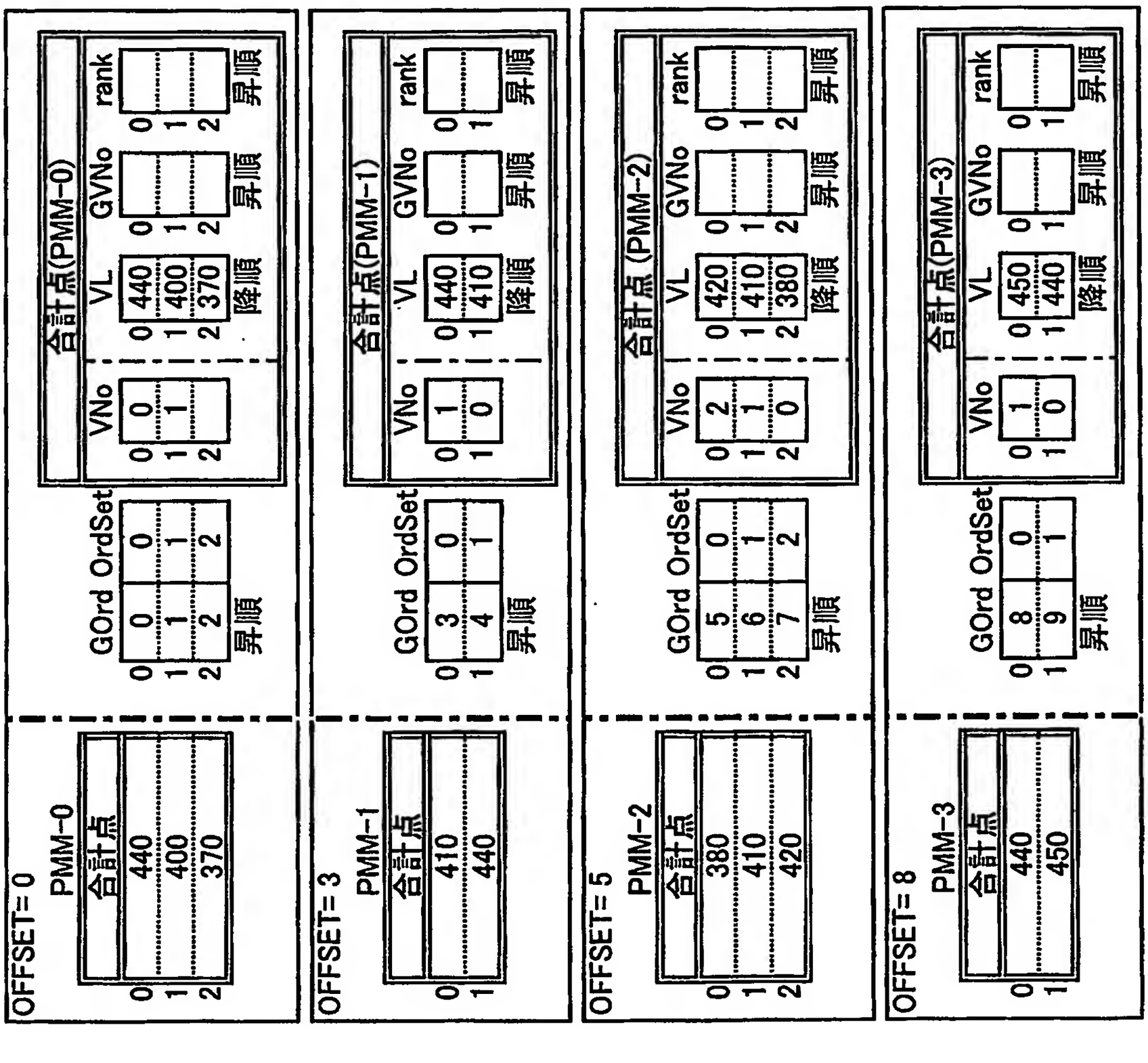
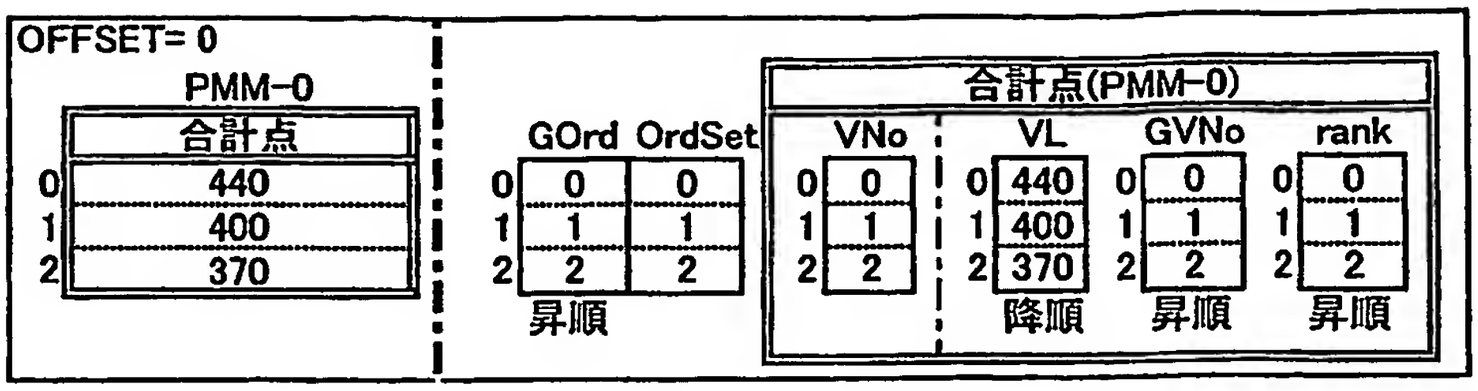

OFFSET= 0
PMM-0
合計点
0 440
1 400
2 370
GOrd OrdSet
0 0 0
1 1 1
2 2 2
昇順
合計点(PMM-0)
VNo VL GVNo
0 0 0 440 0
1 1 1 400 1
2 2 2 370 2
降順 昇順
OFFSET= 3
PMM-1
合計点
0 410
1 440
GOrd OrdSet
0 3 0
1 4 1
昇順
合計点(PMM-1)
VNo VL GVNo
0 1 0 440 0
1 0 1 410 1
降順 昇順
OFFSET= 5
PMM-2
合計点
0 380
1 410
2 420
GOrd OrdSet
0 5 0
1 6 1
2 7 2
昇順
合計点 (PMM-2)
VNo VL GVNo
0 2 0 420 0
1 1 1 410 1
2 0 2 380 2
降順 昇順
OFFSET= 8
PMM-3
合計点
0 440
1 450
GOrd OrdSet
0 8 0
1 9 1
昇順
合計点(PMM-3)
VNo VL GVNo
0 1 0 450 0
1 0 1 440 1
降順 昇順

図36

【図３７】

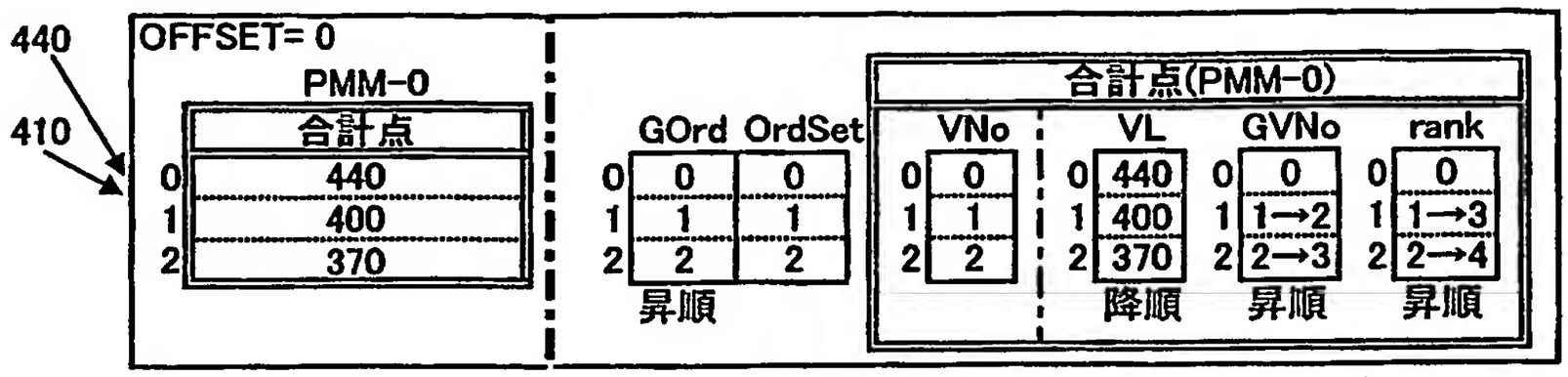
図37

(a)



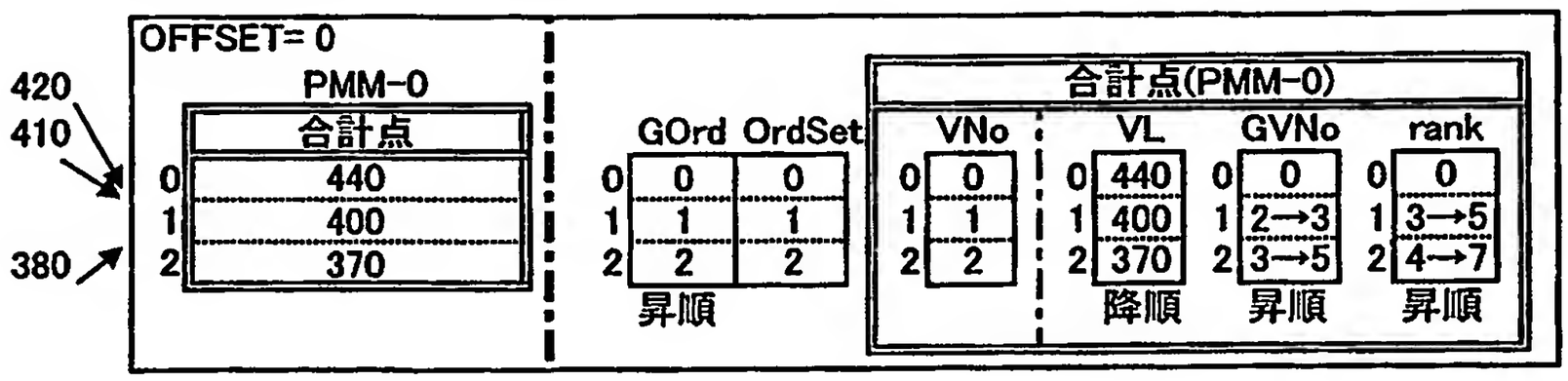
PMM-1から410,440を受信

(b)



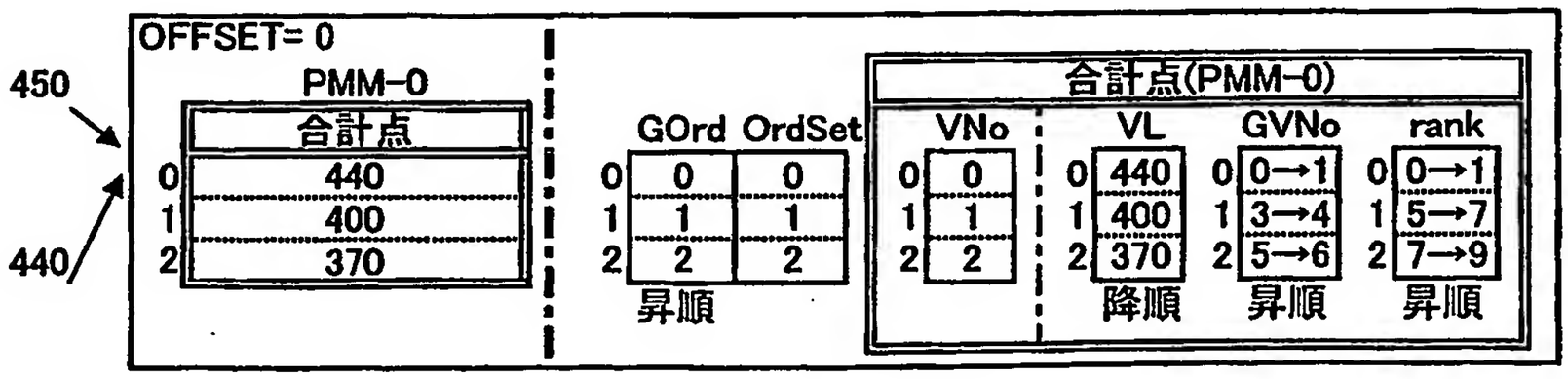
PMM-2から380,410,420を受信

(c)



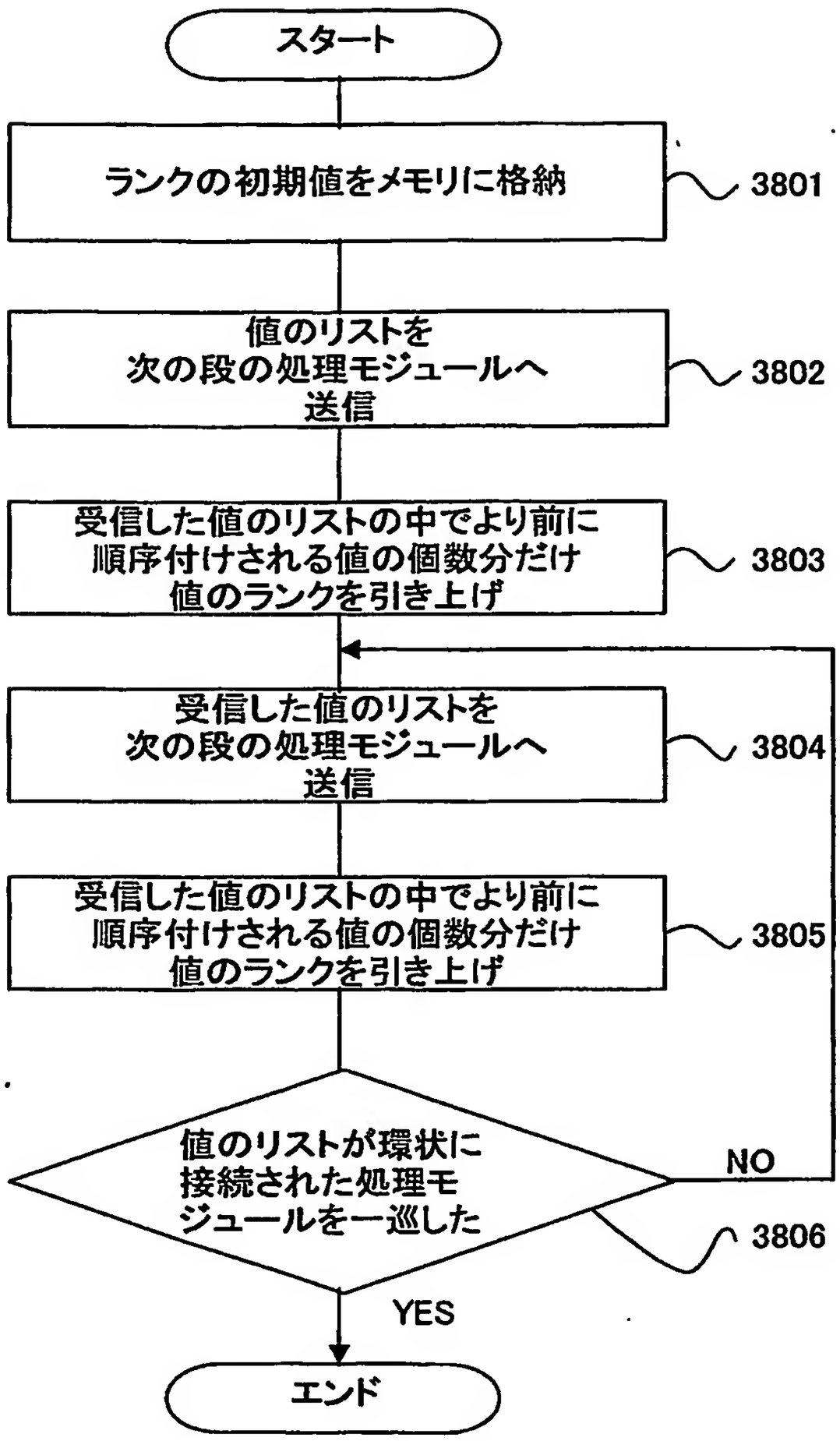
PMM-3から440,450を受信

(d)



【図３８】

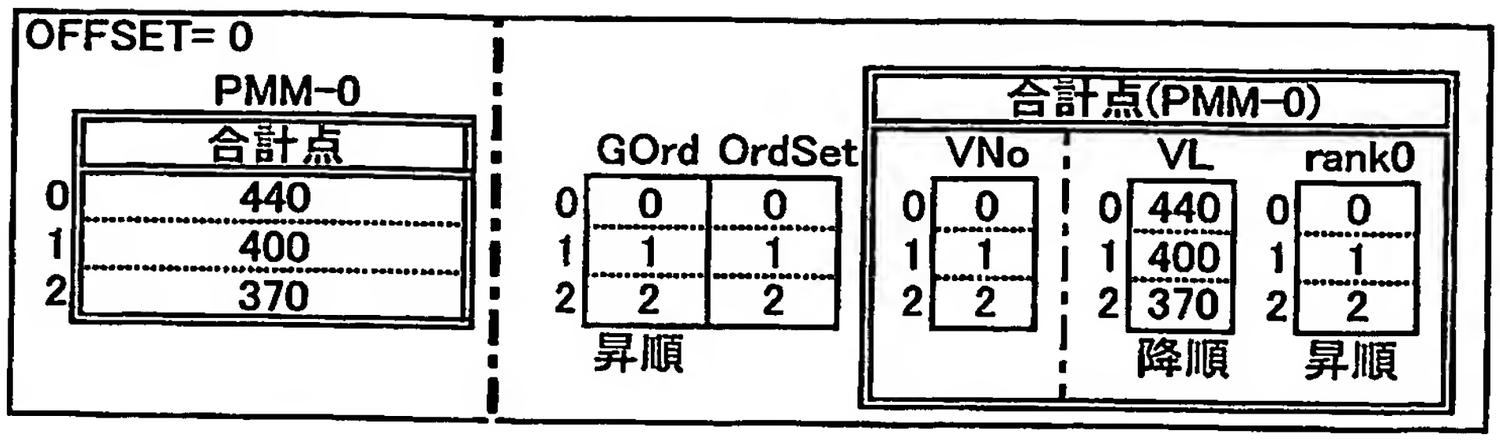
図38


スタート
ランクの初期値をメモリに格納
3801
値のリストを
次の段の処理モジュールへ
送信
3802
受信した値のリストの中でより前に
順序付けされる値の個数分だけ
値のランクを引き上げ
3803
受信した値のリストを
次の段の処理モジュールへ
送信
3804
受信した値のリストの中でより前に
順序付けされる値の個数分だけ
値のランクを引き上げ
3805
値のリストが環状に
接続された処理モ
ジュールを一巡した
NO
3806
YES
エンド

【図３９】

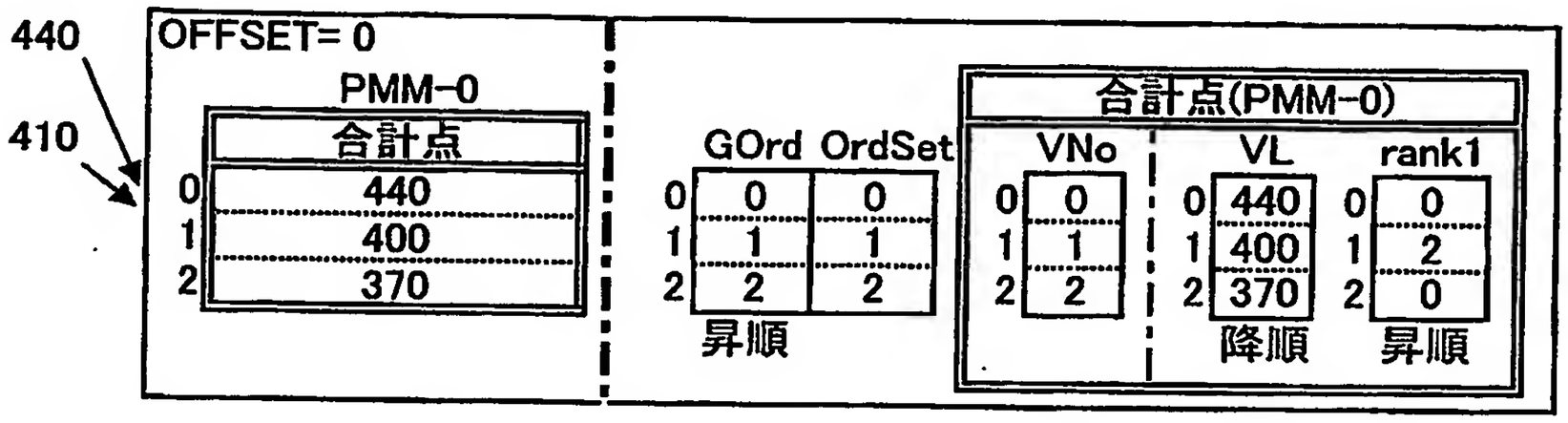
図39

(a)



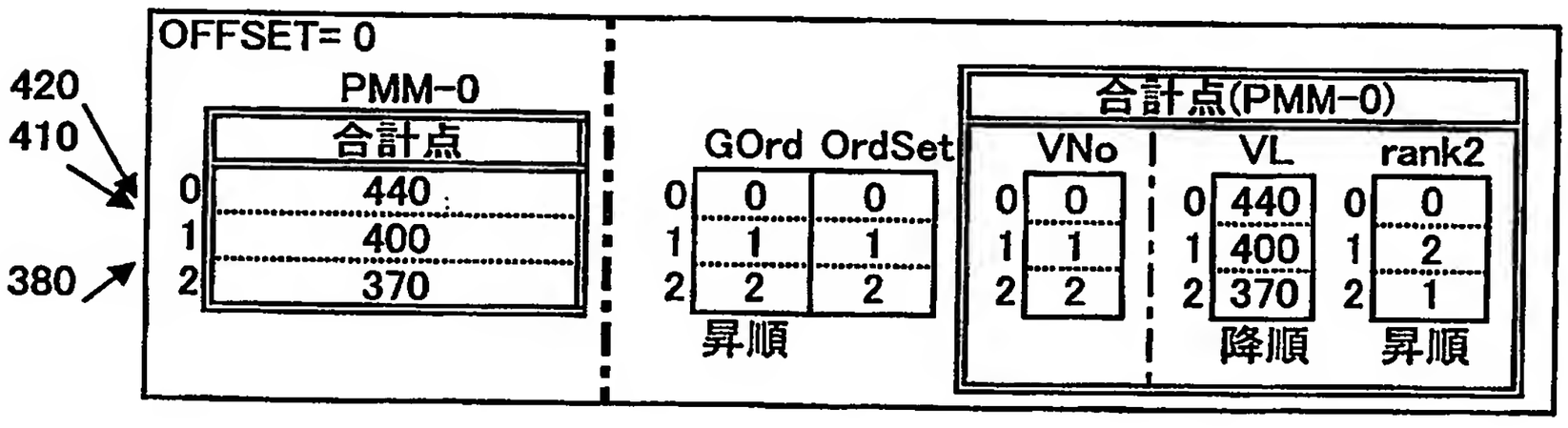
PMM-1から410,440を受信

(b)



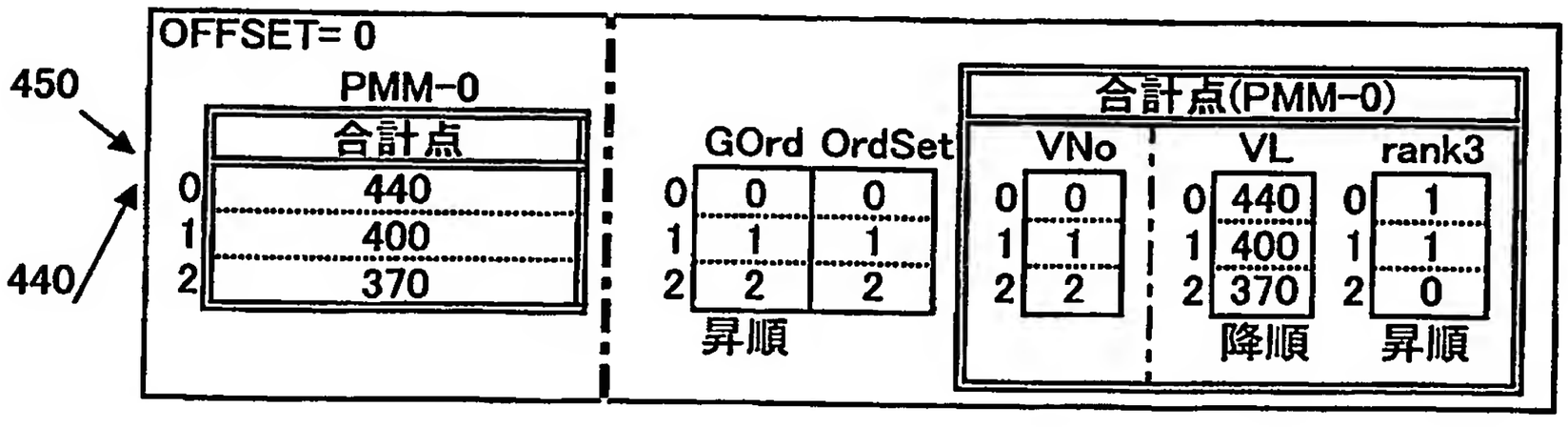
PMM-2から380,410,420を受信

(c)

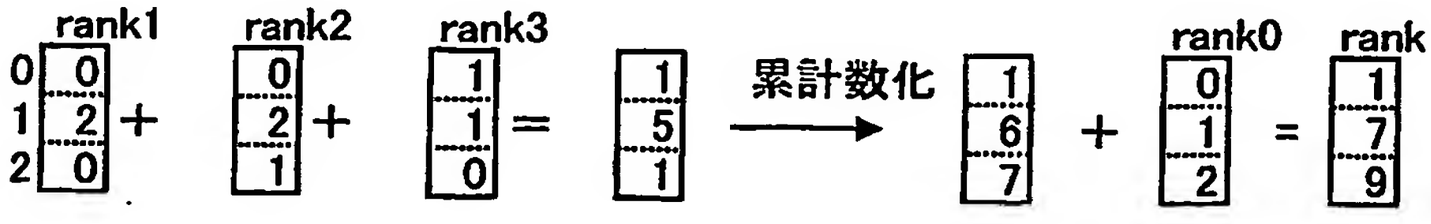


PMM-3から440,450を受信

(d)

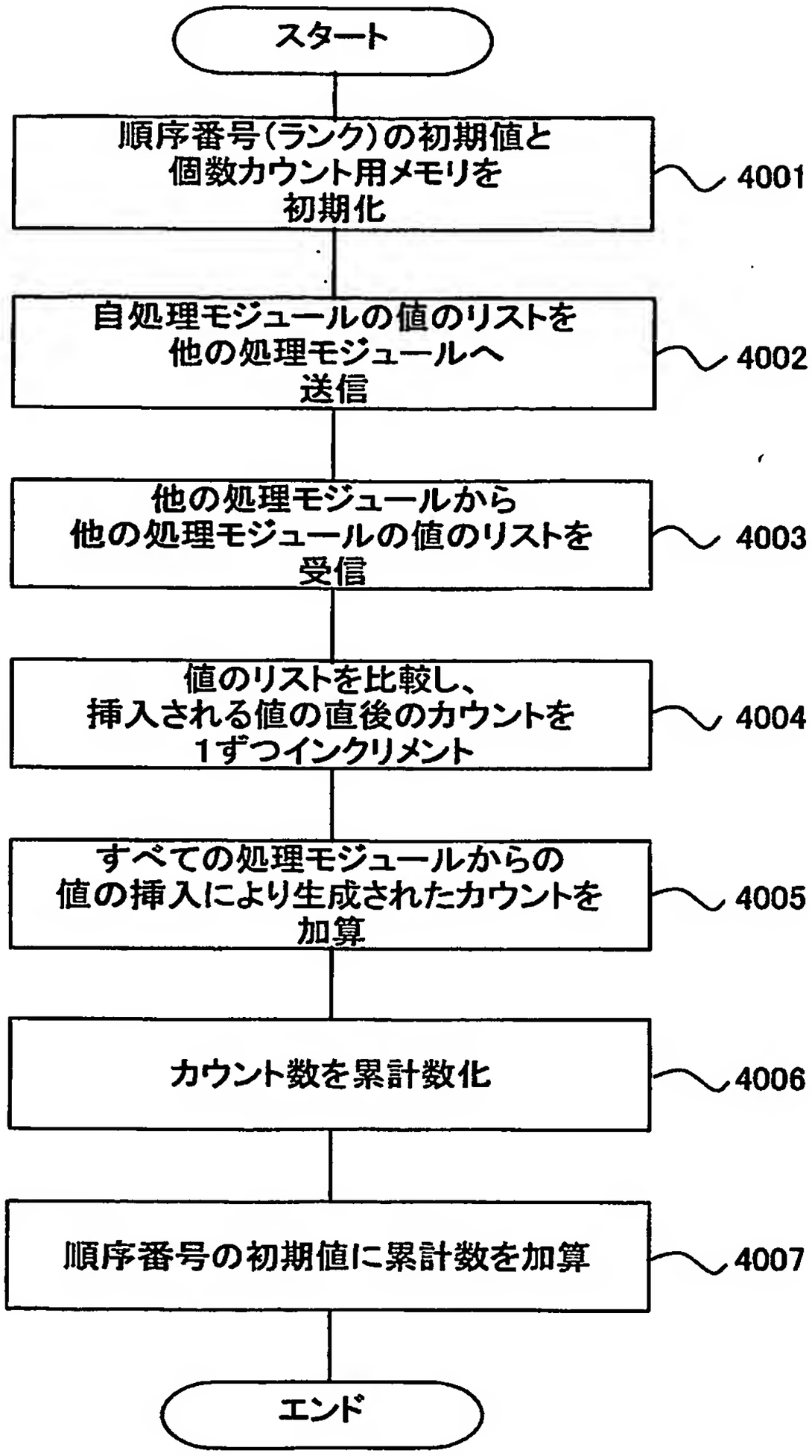


(e)



【図４０】

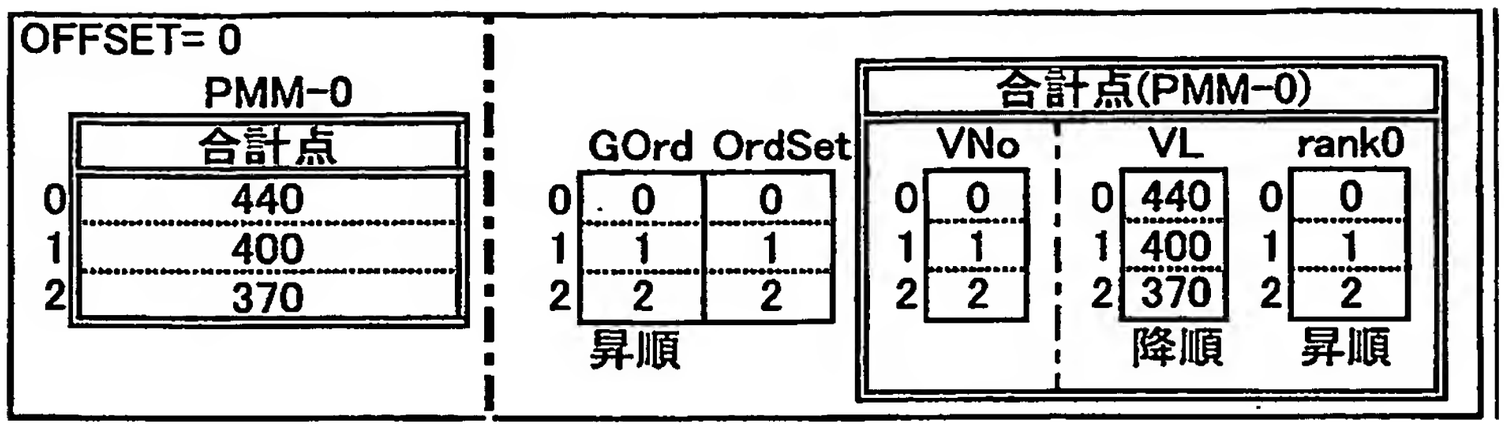
図40


スタート
順序番号(ランク)の初期値と
個数カウント用メモリを
初期化
4001
自処理モジュールの値のリストを
他の処理モジュールへ
送信
4002
他の処理モジュールから
他の処理モジュールの値のリストを
受信
4003
値のリストを比較し、
挿入される値の直後のカウントを
1ずつインクリメント
4004
すべての処理モジュールからの
値の挿入により生成されたカウントを
加算
4005
カウント数を累計数化
4006
順序番号の初期値に累計数を加算
4007
エンド

【図４１】

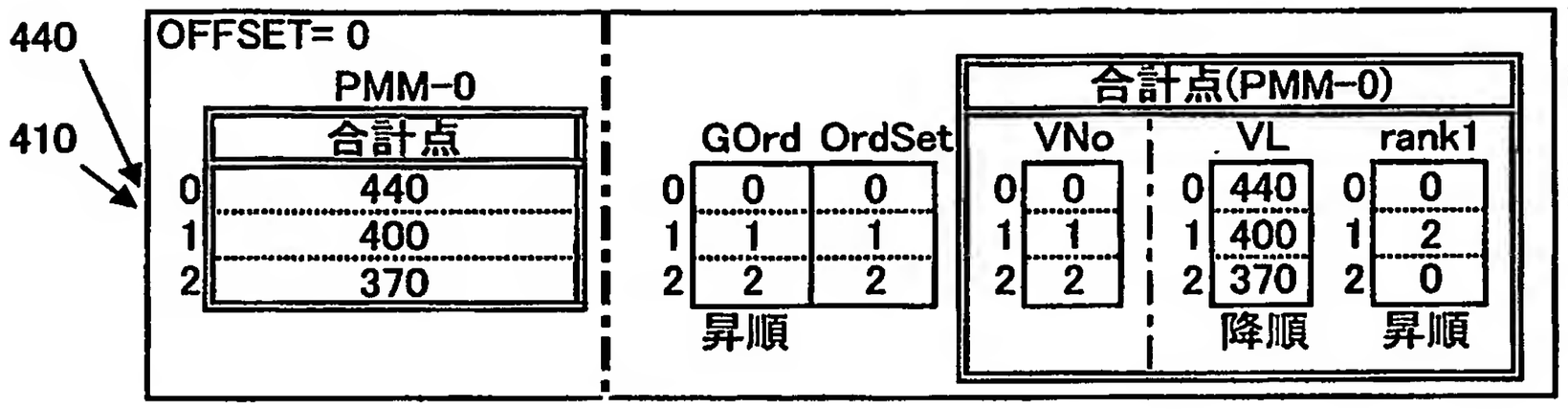
図41

(a)



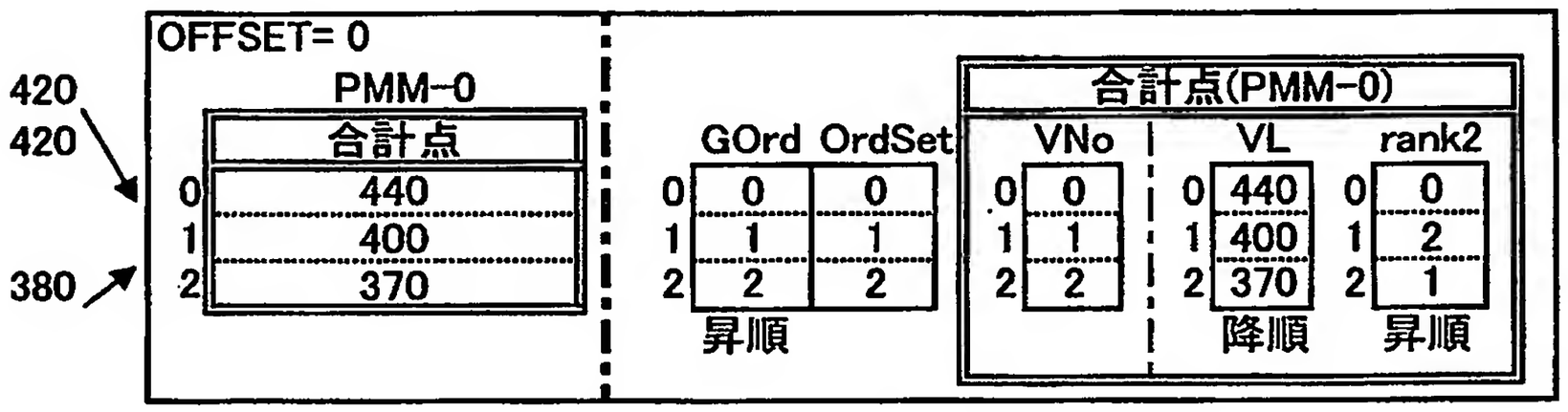
PMM-1から[(410,1),(440,1)]を受信

(b)



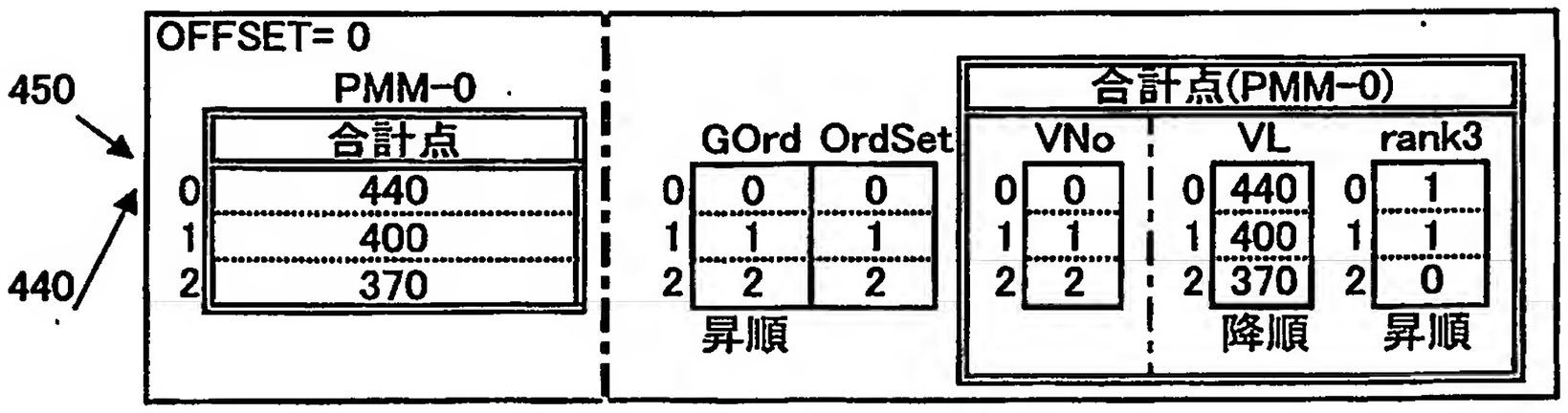
PMM-2から[(380,1),(420,2)]を受信

(c)

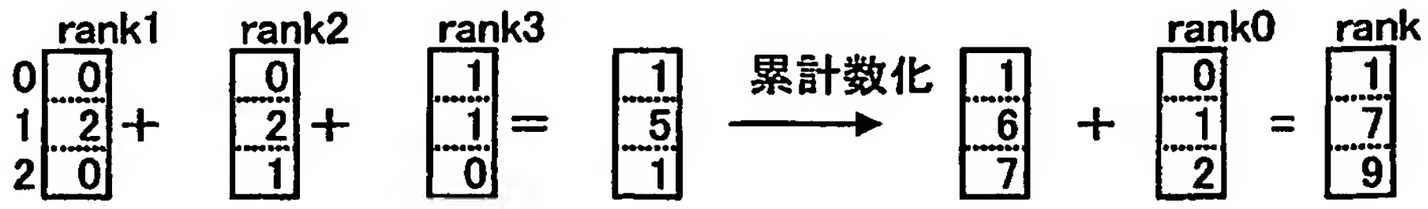


PMM-3から440,450を受信

(d)

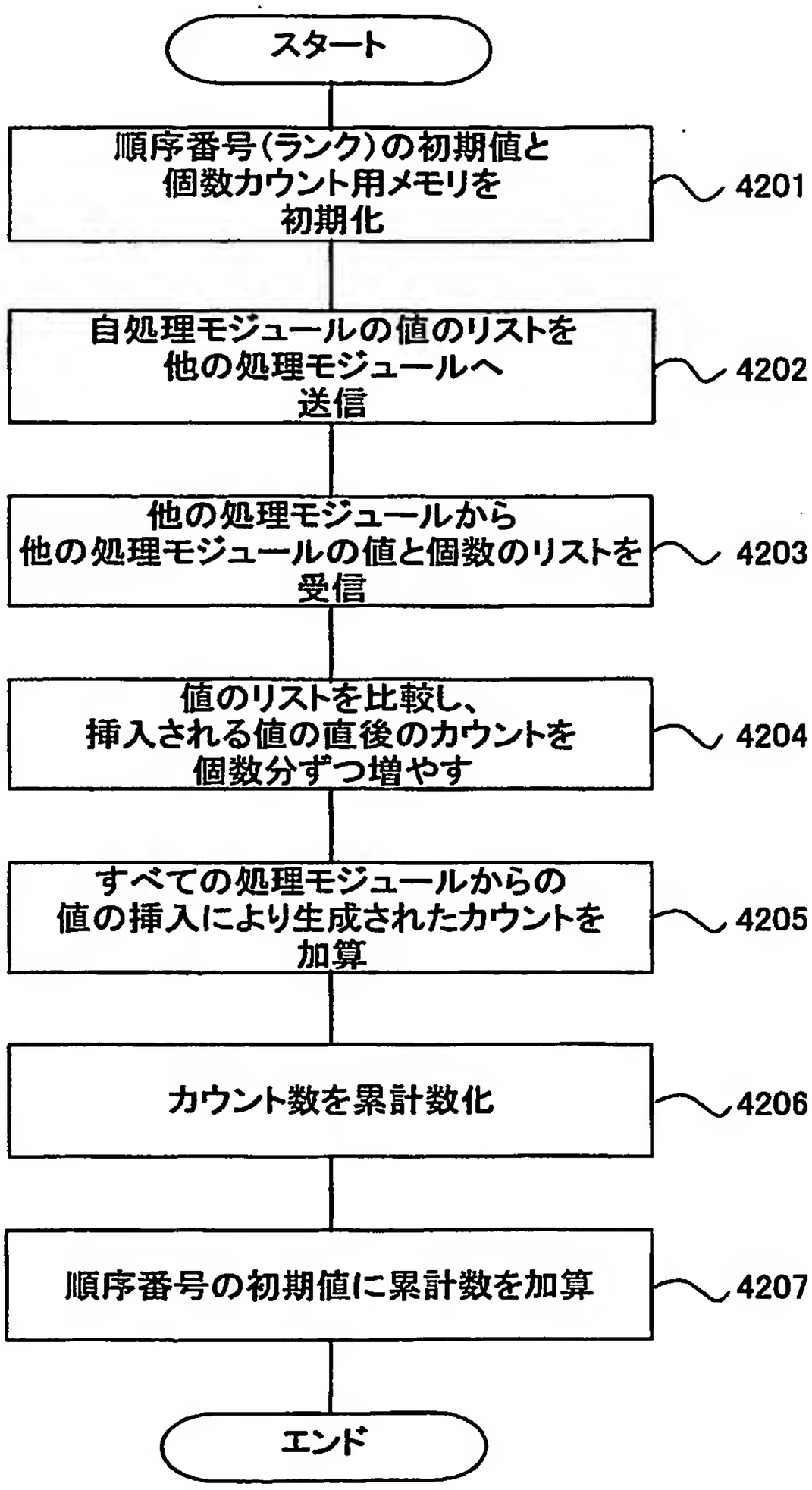


(e)



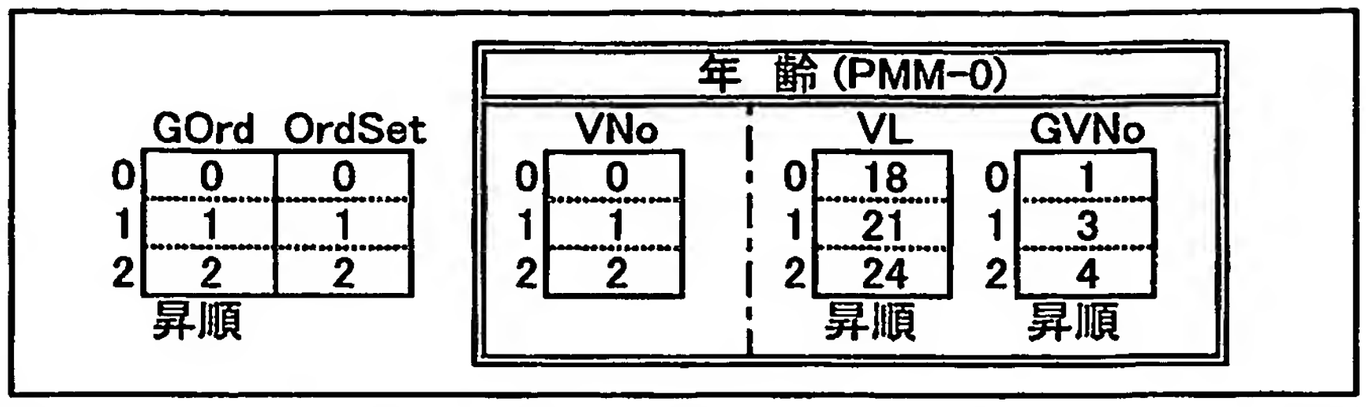
【図４２】

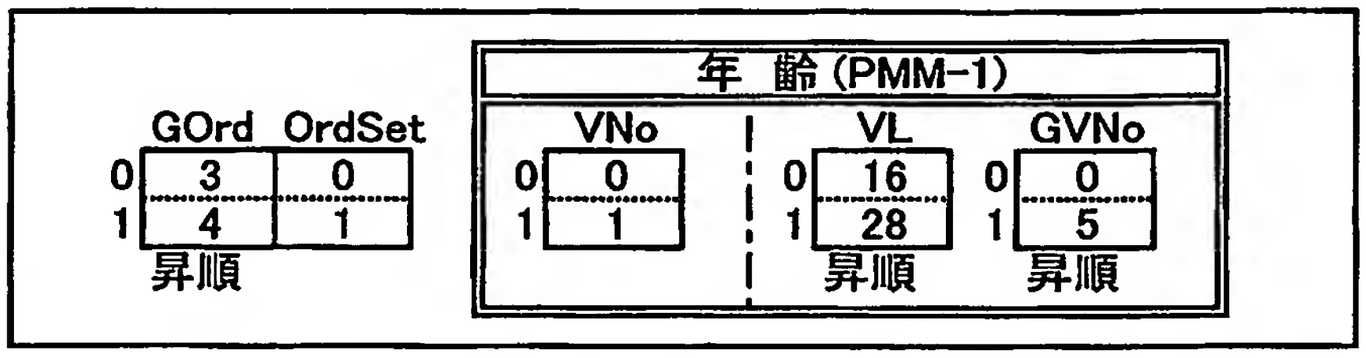
図42

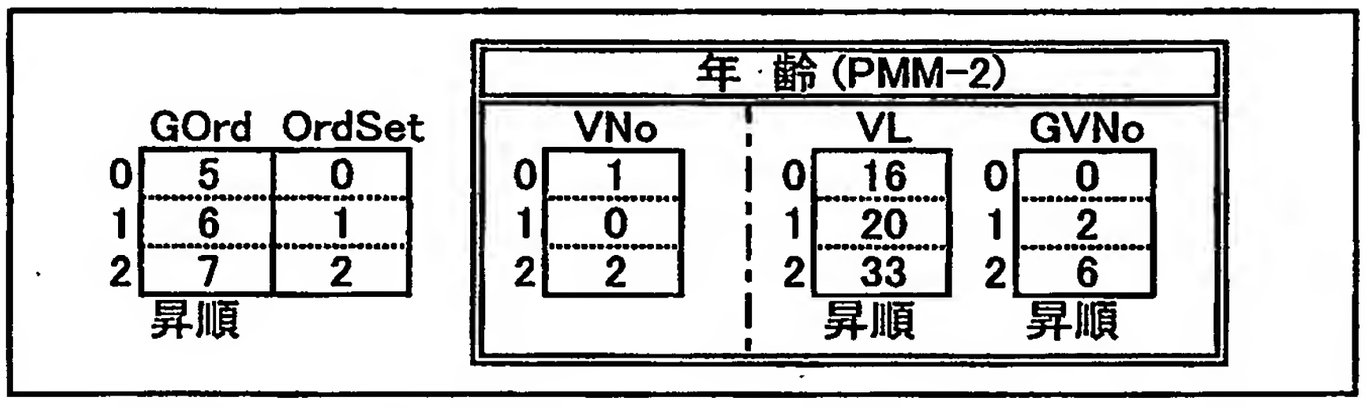

スタート
順序番号(ランク)の初期値と
個数カウント用メモリを
初期化
4201
自処理モジュールの値のリストを
他の処理モジュールへ
送信
4202
他の処理モジュールから
他の処理モジュールの値と個数のリストを
受信
4203
値のリストを比較し、
挿入される値の直後のカウントを
個数分ずつ増やす
4204
すべての処理モジュールからの
値の挿入により生成されたカウントを
加算
4205
カウント数を累計数化
4206
順序番号の初期値に累計数を加算
4207
エンド

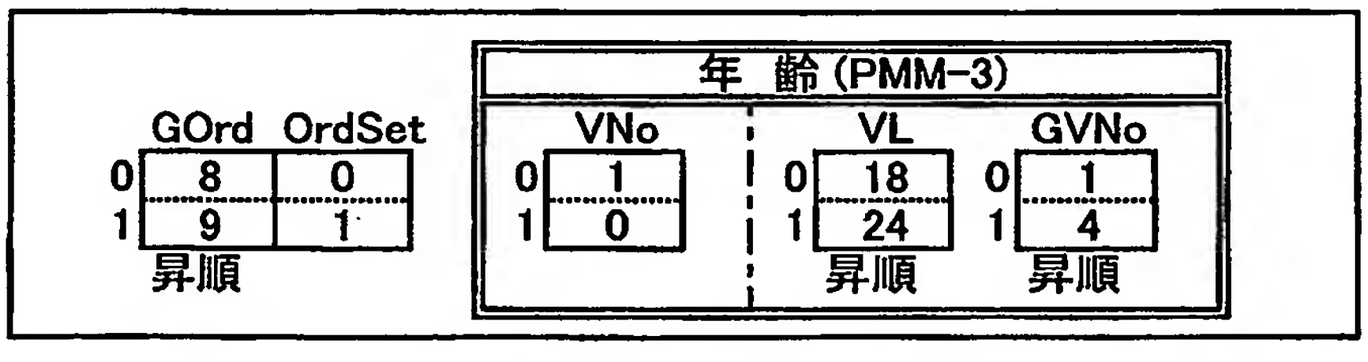
【図４３】

図43


GOrd OrdSet
0 0 0
1 1 1
2 2 2
昇順
年 齢 (PMM-0)
VNo
0 0
1 1
2 2
VL
0 18
1 21
2 24
昇順
GVNo
0 1
1 3
2 4
昇順

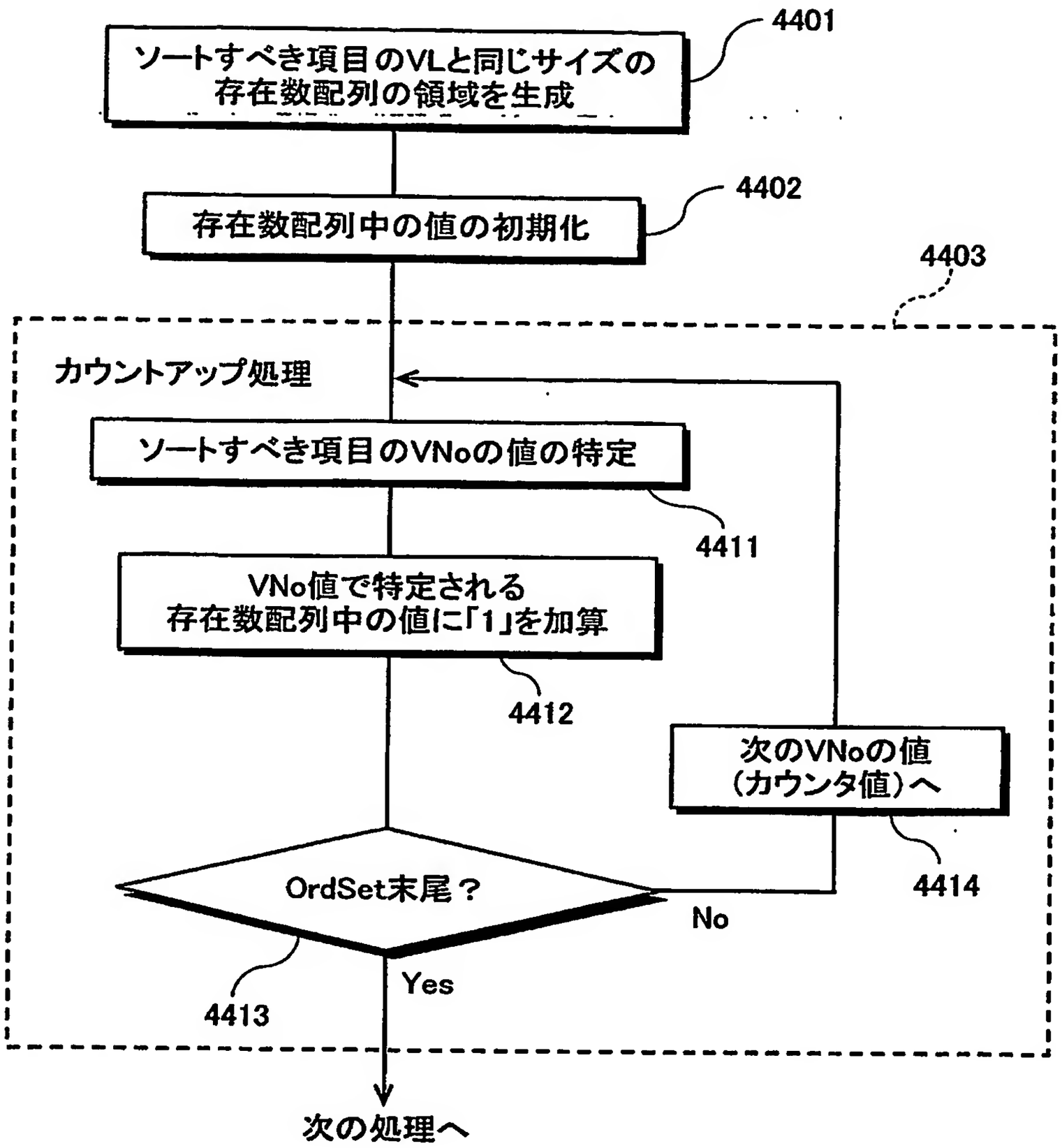

GOrd OrdSet
0 3 0
1 4 1
昇順
年 齢 (PMM-1)
VNo
0 0
1 1
VL
0 16
1 28
昇順
GVNo
0 0
1 5
昇順


GOrd OrdSet
0 5 0
1 6 1
2 7 2
昇順
年 齢 (PMM-2)
VNo
0 1
1 0
2 2
VL
0 16
1 20
2 33
昇順
GVNo
0 0
1 2
2 6
昇順


GOrd OrdSet
0 8 0
1 9 1
昇順
年 齢 (PMM-3)
VNo
0 1
1 0
VL
0 18
1 24
昇順
GVNo
0 1
1 4
昇順

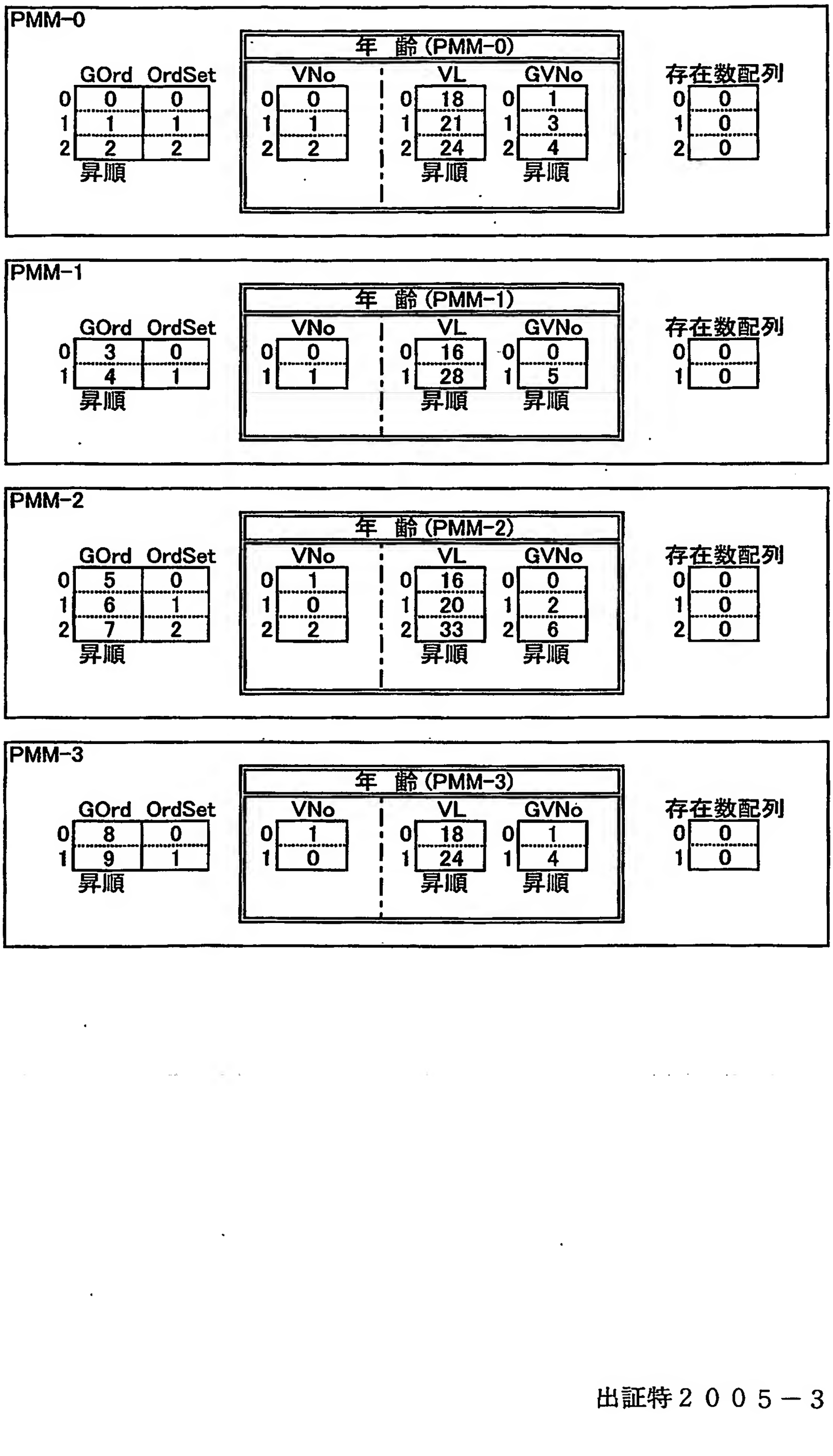
【図４４】

図44


ソートすべき項目のVLと同じサイズの
存在数配列の領域を生成
4401
存在数配列中の値の初期化
4402
4403
カウントアップ処理
ソートすべき項目のVNoの値の特定
4411
VNo値で特定される
存在数配列中の値に「1」を加算
4412
次のVNoの値
（カウンタ値）へ
4414
OrdSet末尾？
No
Yes
4413
次の処理へ

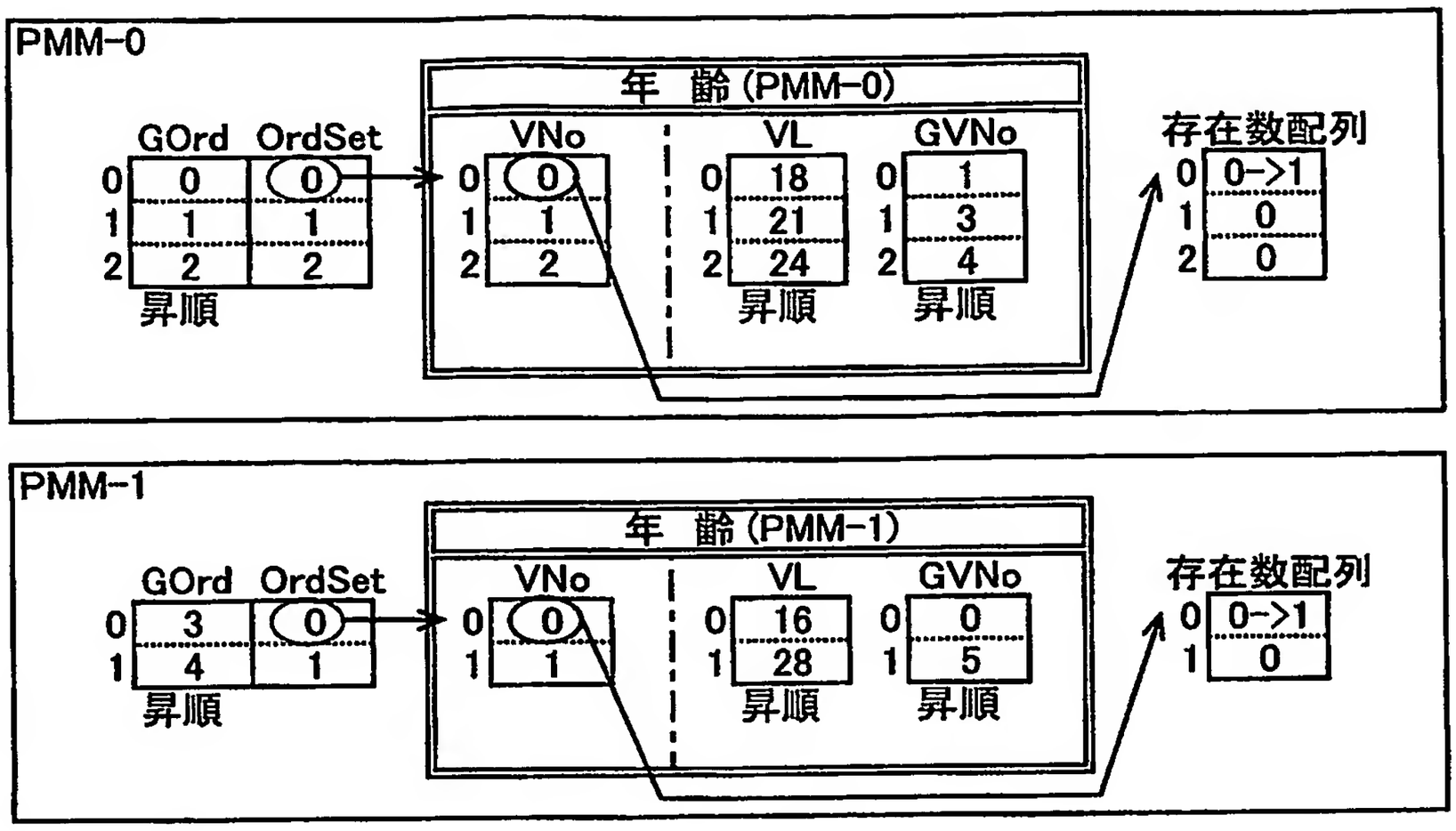
【図４５】

図45


PMM-0
GOrd OrdSet
0 0 0
1 1 1
2 2 2
昇順
年 齢 (PMM-0)
VNo
0 0
1 1
2 2
VL
0 18
1 21
2 24
昇順
GVNo
0 1
1 3
2 4
昇順
存在数配列
0 0
1 0
2 0
PMM-1
GOrd OrdSet
0 3 0
1 4 1
昇順
年 齢 (PMM-1)
VNo
0 0
1 1
VL
0 16
1 28
昇順
GVNo
0 0
1 5
昇順
存在数配列
0 0
1 0
PMM-2
GOrd OrdSet
0 5 0
1 6 1
2 7 2
昇順
年 齢 (PMM-2)
VNo
0 1
1 0
2 2
VL
0 16
1 20
2 33
昇順
GVNo
0 0
1 2
2 6
昇順
存在数配列
0 0
1 0
2 0
PMM-3
GOrd OrdSet
0 8 0
1 9 1
昇順
年 齢 (PMM-3)
VNo
0 1
1 0
VL
0 18
1 24
昇順
GVNo
0 1
1 4
昇順
存在数配列
0 0
1 0

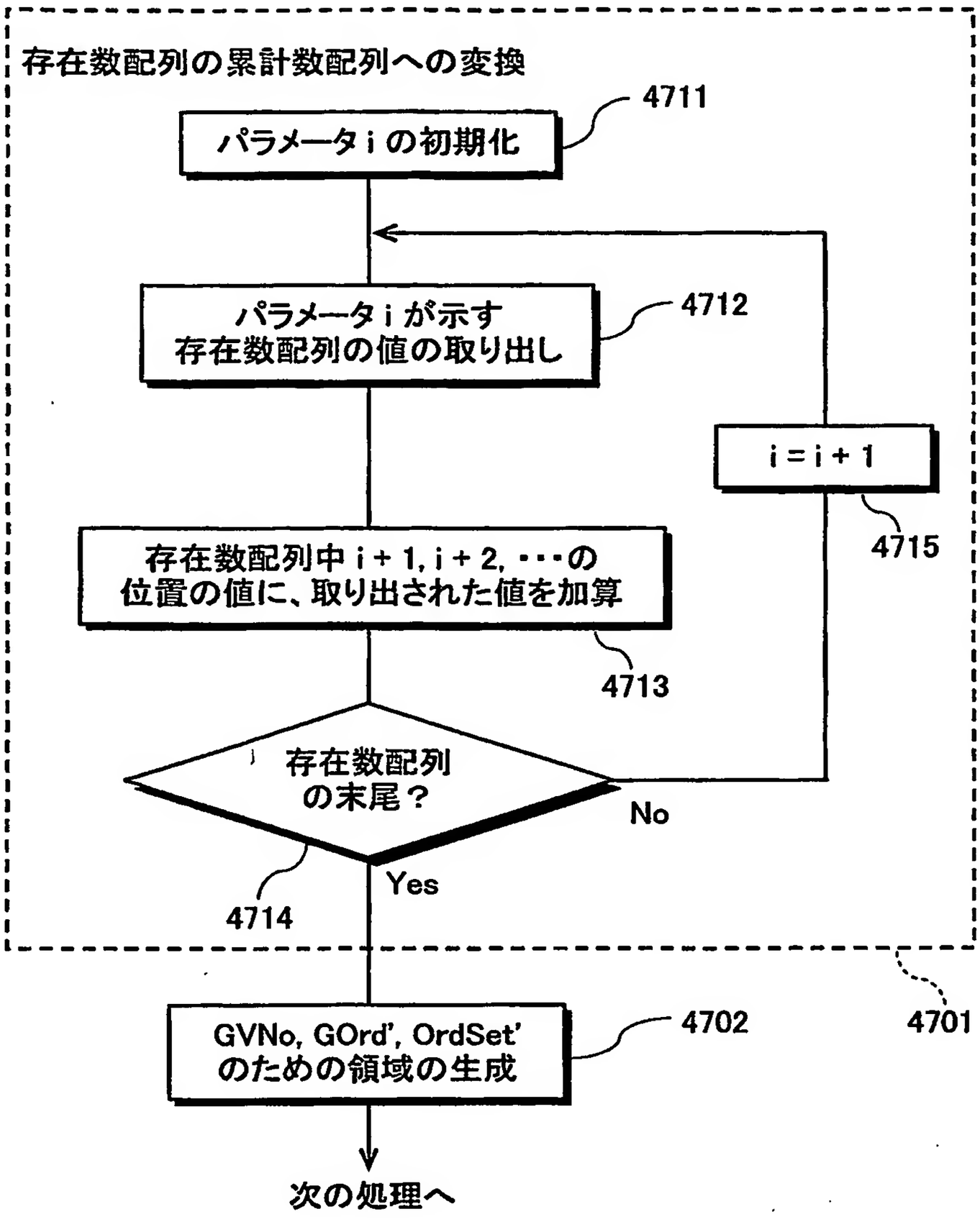
【図４６】

図46

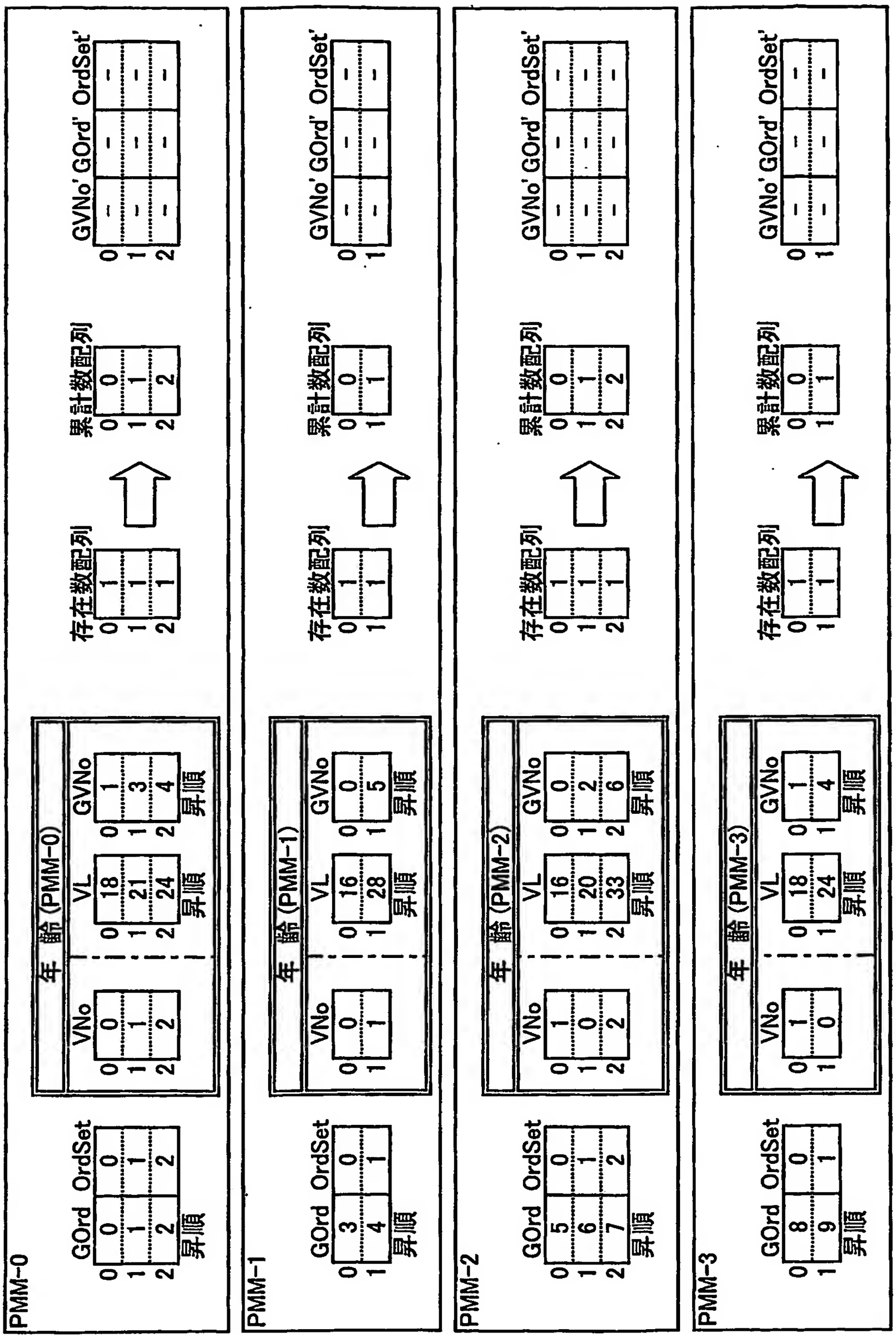

PMM-0
年　齢(PMM-0)
GOrd
OrdSet
VNo
VL
GVNo
存在数配列
昇順
PMM-1
年　齢(PMM-1)
PMM-2
年　齢(PMM-2)
PMM-3
年　齢(PMM-3)

【図４７】

図47

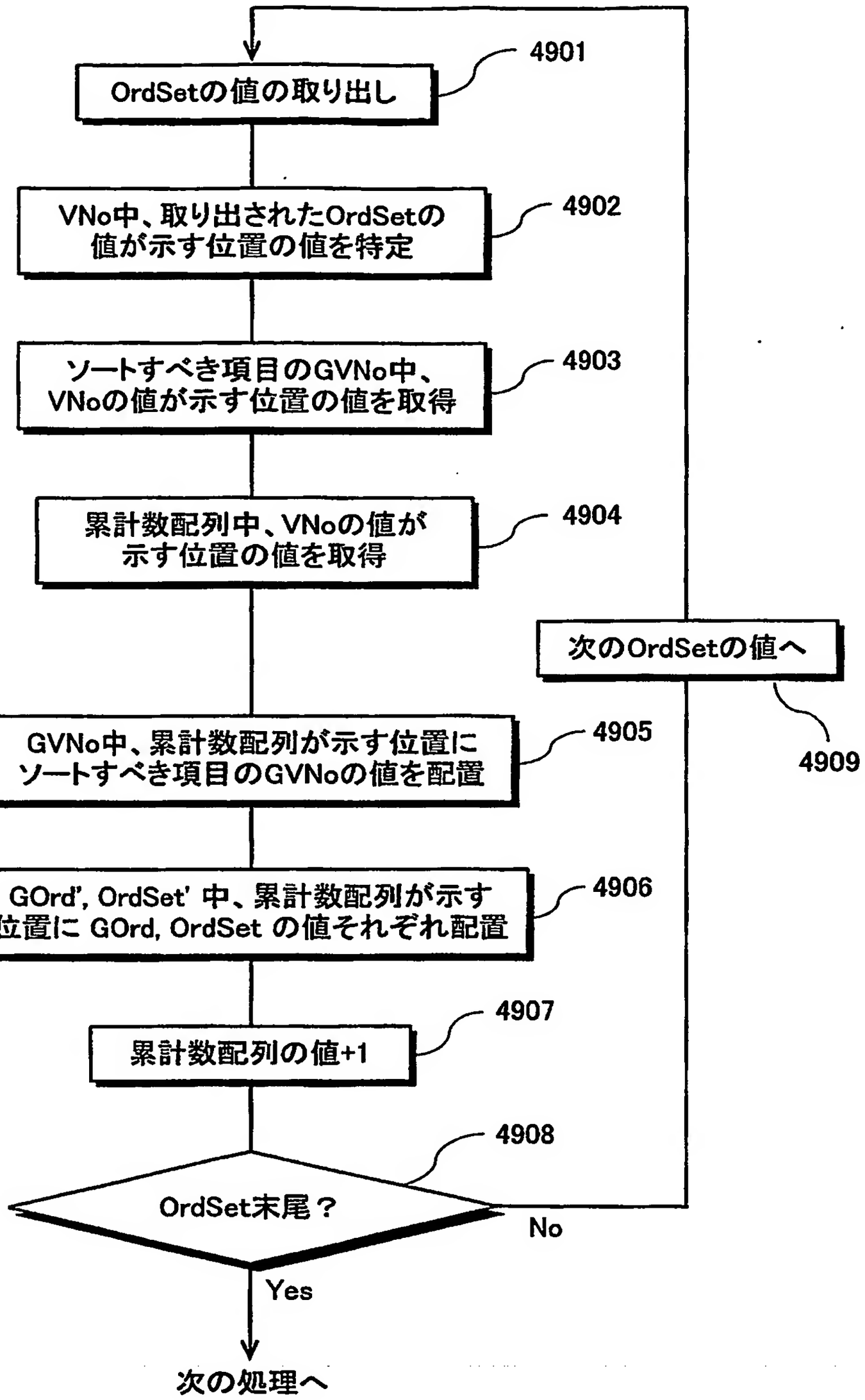

存在数配列の累計数配列への変換
パラメータｉの初期化
4711
パラメータｉが示す
存在数配列の値の取り出し
4712
存在数配列中ｉ＋１，ｉ＋２，・・・の
位置の値に、取り出された値を加算
4713
存在数配列
の末尾？
No
Yes
4714
ｉ＝ｉ＋１
4715
4701
GVNo, GOrd', OrdSet'
のための領域の生成
4702
次の処理へ

【図４８】


PMM-0
PMM-1
PMM-2
PMM-3
年 齢 (PMM-0)
年 齢 (PMM-1)
年 齢 (PMM-2)
年 齢 (PMM-3)
VNo
VL
GVNo
昇順
累計数配列
GVNo' GOrd' OrdSet'
存在数配列
GOrd OrdSet
図48

【図４９】

図49


OrdSetの値の取り出し
4901
VNo中、取り出されたOrdSetの値が示す位置の値を特定
4902
ソートすべき項目のGVNo中、VNoの値が示す位置の値を取得
4903
累計数配列中、VNoの値が示す位置の値を取得
4904
次のOrdSetの値へ
4909
GVNo中、累計数配列が示す位置にソートすべき項目のGVNoの値を配置
4905
GOrd', OrdSet' 中、累計数配列が示す位置に GOrd, OrdSet の値それぞれ配置
4906
累計数配列の値+1
4907
OrdSet末尾？
4908
No
Yes
次の処理へ

【図５０】

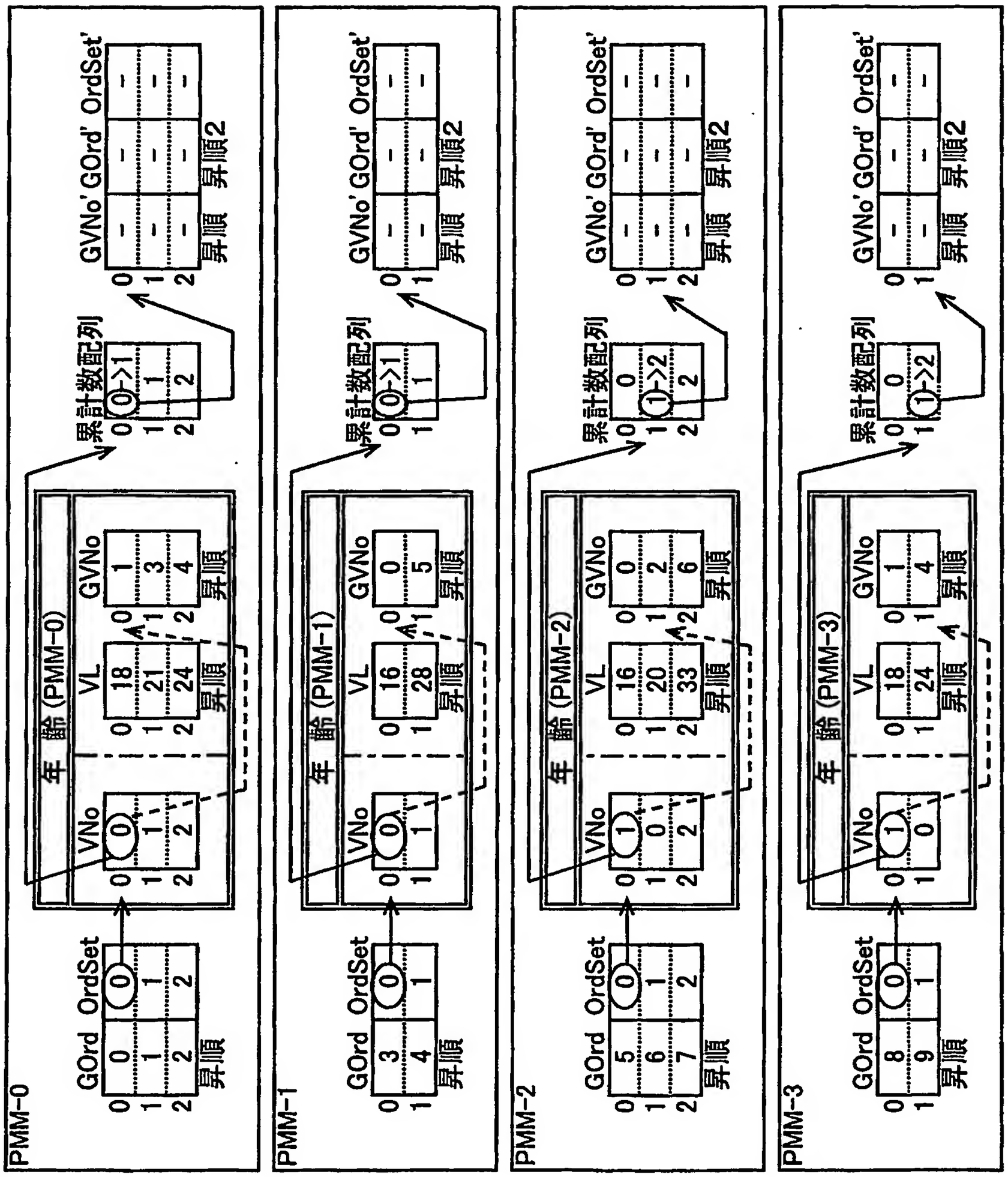

PMM-0
PMM-1
PMM-2
PMM-3
GOrd OrdSet
昇順
年 齢 (PMM-0)
年 齢 (PMM-1)
年 齢 (PMM-2)
年 齢 (PMM-3)
VNo
VL
GVNo
昇順
累計数配列
GVNo' GOrd' OrdSet'
昇順 昇順2

図50

【図５１】

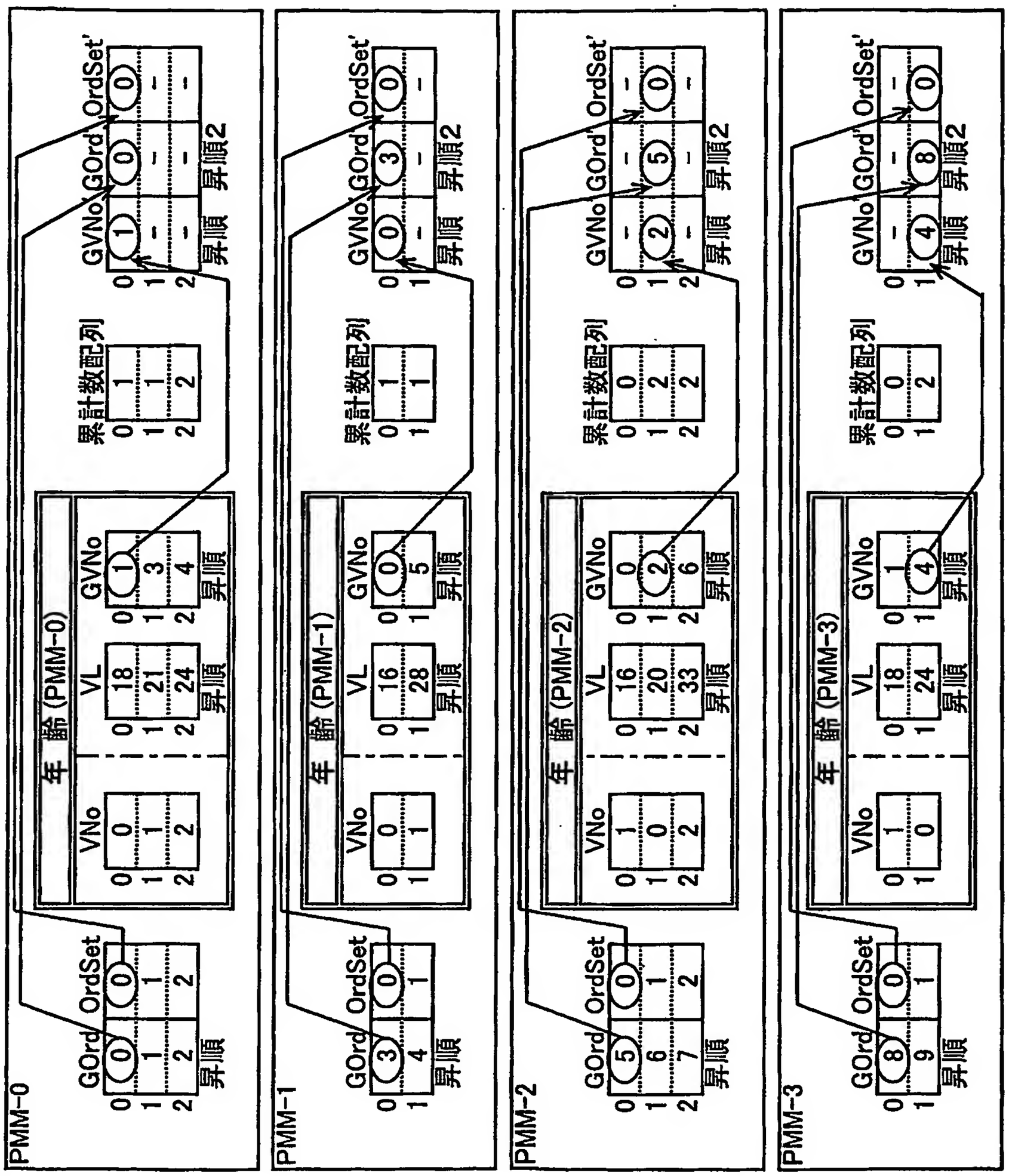

PMM-0
PMM-1
PMM-2
PMM-3
年　齢 (PMM-0)
年　齢 (PMM-1)
年　齢 (PMM-2)
年　齢 (PMM-3)
VNo
VL
GVNo
累計数配列
GOrd
OrdSet
昇順
昇順2

図51

【図５２】

図52

PMM-0

累計数配列

0
1
2

	GVNo'	GOrd'	OrdSet'
0	1	0	0
1	3	1	1
2	4	2	2

昇順　昇順2

PMM-1

累計数配列

0
1

	GVNo'	GOrd'	OrdSet'
0	0	3	0
1	5	4	1

昇順　昇順2

PMM-2

累計数配列

0
1
2

	GVNo'	GOrd'	OrdSet'
0	0	6	1
1	2	5	0
2	6	7	2

昇順　昇順2

PMM-3

累計数配列

0
1

	GVNo'	GOrd'	OrdSet'
0	1	9	1
1	4	8	0

昇順　昇順2

【図５３】

図53

(A)　配列作成

OrdSet	性別のGVNo	年齢のGVNo	GOrd
0	1	2	0
1	0	0	1
2	1	1	2

(B)　順序番号初期化

OrdSet	順序番号
0	0
1	0
2	0

(C)　順序番号付与1

OrdSet	順序番号
0	0→1
1	0
2	0→1

(D)　順序番号付与2

OrdSet	順序番号
0	1→2
1	0
2	1

(E)　結果

順序番号	OrdSet	性別のGVNo	年齢のGVNo	GOrd
2	0	1	2	0
0	1	0	0	1
1	2	1	1	2

(F)　結果

順序番号	OrdSet	性別のGVNo	年齢のGVNo	GOrd
0	1	0	0	1
1	2	1	1	2
2	0	1	2	0

【書類名】要約書
【要約】
【課題】　並列コンピュータのアーキテクチャを採用して大量のデータを情報処理する際に、複数の処理モジュール間でのデータ処理を少ない通信量で高速に実現する。
【解決手段】　本発明によれば、各処理モジュールは、自処理モジュールのメモリに格納されている値のリストである第１のリストを情報処理システム内の他の処理モジュールへ送信し、他の処理モジュールから自処理モジュールへ送信された値のリストである少なくとも一つの第２のリストを受信し、第２のリスト中の値と第１のリスト中の値を比較し、第２のリスト中の値が第１のリスト中の値と一致した場合に、第１のリスト中の一致した値に対応したカウンタを１ずつ増やす。
【選択図】　図３４

認定・付加情報

特許出願の番号	特願２００３－４３１２４８
受付番号	５０３０２１３８４３８
書類名	特許願
担当官	末武　実　　１９１２
作成日	平成１５年１２月２６日

＜認定情報・付加情報＞

【提出日】　平成15年12月25日

特願２００３－４３１２４８

出　願　人　履　歴　情　報

識別番号　［５９８１０８５１５］

１．変更年月日　１９９８年　８月１１日
［変更理由］　新規登録
住　所　神奈川県横浜市神奈川区松見町４丁目１１０１番地７　コート
ハウス菊名８０４号
氏　名　古庄　晋二

出証番号　出証特２００５－３００９１２５

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/JP04/019181

International filing date: 22 December 2004 (22.12.2004)

Document type: Certified copy of priority document

Document details: Country/Office: JP
Number: 2003-431248
Filing date: 25 December 2003 (25.12.2003)

Date of receipt at the International Bureau: 24 February 2005 (24.02.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)